IBM

Setting up the application serving environment

Note

Before using this information, be sure to read the general information under “Notices” on page 473.

Compilation date: May 5, 2006

Contents

How to send your comments

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

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

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

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

Chapter 1. Configuring ports

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

For more information about port numbers that your iSeries system currently uses, enter the NETSTAT *CNN command on the command line. Press **F14** to view assigned port numbers.

You can also use the port validator tool to find port conflicts between different WebSphere Application Server profiles, products, and servers. See the information center.

1. Review the port number settings, especially when you are planning to coexist.

   You can use the **dspwasinst** command-line tool to display the port information for a profile. See the information center.
2. **Optional:** Change the port number settings.

   You can set port numbers when configuring the product after installation.
   - During profile creation using the **manageprofiles** command, you can accept the default port values or you can specify your port settings. If you want to specify ports, you can do so in any of the following ways:
     – Specify the use of a port file that contains the port values.
     – Specify the use of a starting port value.
     – Specify the use of the default port values.
   - You can use the **chgwassvr** command to change the ports for an application server within a profile.

Port number settings in WebSphere Application Server versions

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

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

Notes:

- When you install WebSphere Application Server, the default instance is created with the default port values. When you create a WebSphere Application Server instance with the crtwasinst script, you can specify different port values.
- For more information about port numbers that your iSeries system currently uses, enter the NETSTAT *CNN command on the CL command line. Press F14 to view assigned port numbers.
- You can also use the port validator tool to find port conflicts between different WebSphere Application Server profiles, products, and servers. See the *Using the administrative clients* PDF for more information.

Version 6.1 port numbers

Table 1. Port definitions for WebSphere Application Server Version 6.1

Port Name	Default Value		Files
	Application Server	Deployment Manager	
HTTP_Transport Port (WC_defaulthost)	9080	9080	serverindex.xml and virtualhosts.xml
Administrative Console Port (WC_adminhost)	9060	9060	
HTTPS Transport Port (WC_defaulthost_secure)	9443	9443	
Administrative Console Secure Port (WC_adminhost_secure)	9043	9043	
Bootstrap Port (BOOTSTRAP_ADDRESS)	2809	9809	serverindex.xml
SOAP Connector Port (SOAP_CONNECTOR_ADDRESS)	8880	8879	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9401	9401	
CSIV2 Server Authentication Port (CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS)	9403	9403	
CSIV2 Multi-authentication Listener Port (CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS)	9402	9402	
ORB Listener Port (ORB_LISTENER_ADDRESS)	9100	9100	
High Availability Manager Communication Port (DCS_UNICAST_ADDRESS)	9353	9352	
Service Integration Port (SIB_ENDPOINT_ADDRESS)	7276	7276	
Service Integration Secure Port (SIB_ENDPOINT_SECURE_ADDRESS)	7286	7286	
MQ Transport Port (SIB_MQ_ENDPOINT_ADDRESS)	5558	Not applicable	
MQ Transport Secure Port (SIB_MQ_ENDPOINT_SECURE_ADDRESS)	5578	Not applicable	
SIP Container Port (SIP_DEFAULTHOST)	5060	Not applicable	
SIP Container Secure Port (SIP_DEFAULTHOST_SECURE)	5061	Not applicable	
Internal JMS Server Port (JMSSERVER_SECURITY_PORT)	5557	Not applicable	
DRS_CLIENT_ADDRESS **Deprecation:** This port is deprecated and is no longer used in the current version of WebSphere Application Server.	7873	7989	
IBM HTTP Server Port	80	Not applicable	virtualhosts.xml, plugin-cfg.xml, and *web_server_root*/conf/ httpd.conf
IBM HTTP Server Administration Port	2001	Not applicable	serverindex.xml
Cell Discovery Address (CELL_DISCOVERY_ADDRESS)	Not applicable	7277	
Node Discovery Address (NODE_ DISCOVERY_ADDRESS)	Not applicable	Not applicable	
Node Multicast Discovery Address (NODE_MULTICAST_DISCOVERY_ADDRESS)	Not applicable	Not applicable	
Node IPV6 Discovery Address (NODE_IPV6_MULTICAST_DISCOVERY_ADDRESS)	Not applicable	Not applicable	

When you federate an application server node into a deployment manager cell, the deployment manager instantiates the node agent server process on the application server node. The node agent server uses these port assignments by default.

Table 2. Port definitions for the Version 6.1 node agent server process

Port Name	Default Value	File
Bootstrap Port (BOOTSTRAP_ADDRESS)	2809	serverindex.xml
SOAP Connector Port (SOAP_CONNECTOR_ADDRESS)	8879	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9901	
CSIV2 Server Authentication Port (CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS)	9201	
CSIV2 Multi-authentication Listener Port (CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS)	9202	
ORB Listener Port (ORB_LISTENER_ADDRESS)	9100	
High Availability Manager Communication Port (DCS_UNICAST_ADDRESS)	9353	
Node Discovery Address (NODE_ DISCOVERY_ADDRESS)	7272	
Node Multicast Discovery Address (NODE_MULTICAST_DISCOVERY_ADDRESS)	5000	
Node IPV6 Discovery Address (NODE_IPV6_MULTICAST_DISCOVERY_ADDRESS)	5001	

Version 6.0.x port numbers

Table 3. Port definitions for WebSphere Application Server Version 6.0.x

Port Name	Default Value		Files
	Application Server	Deployment Manager	
Web Container Port (WC_defaulthost)	9080	Not applicable	server.xml, plugin-cfg.xml, and virtualhosts.xml
Web Container Secure Port (WC_defaulthost_secure) **Note:** If you change this port number, remember the following information: • To use secure (SSL-enabled) ports you must have the OS/400 or i5/OS Digital Certificate Manager product (5722SS1 option 34) and a Cryptographic Access Provider product (such as 5722AC3) installed. • If you change this port, you must regenerate the Web server plug-in configuration for the application server.	9443	Not applicable	
Administrative Console Port (WC_adminhost)	9060	9060	server.xml and virtualhosts.xml
Administrative Console Secure Port (WC_adminhost_secure)	9043	9043	
Name Service or RMI Connector Port (BOOTSTRAP_ADDRESS)	2809	9809	serverindex.xml
SOAP Port (SOAP_CONNECTOR_ADDRESS)	8880	8879	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9501	9401	
Common Secure Interoperability Version 2 (CSIV2) Server Transport Port (CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS)	9503	9403	
CSIV2 Client Transport Port (CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS)	9502	9402	
Object Request Broker (ORB) Listener Port (ORB_LISTENER_ADDRESS)	Not applicable	9100	
Java Message Service (JMS) Queued Port (JMSSERVER_QUEUED_ADDRESS)	5558	Not applicable	
JMS Direct Port (JMSSERVER_DIRECT_ADDRESS)	5559	Not applicable	
JMS Security Port (JMSSERVER_SECURITY_PORT)	5557	Not applicable	
Data Replication Service Client Port (DRS_CLIENT_ADDRESS) **Deprecation:** This port is deprecated and is no longer used in the current version of WebSphere Application Server.	7873	7989	
IBM HTTP Server Port	80	Not applicable	virtualhosts.xml, plugin-cfg.xml, and *web_server_root*/conf/httpd.conf

Table 3. Port definitions for WebSphere Application Server Version 6.0.x (continued)

Port Name	Default Value		Files
	Application Server	Deployment Manager	
IBM HTTP Server Administration Port	2001	Not applicable	serverindex.xml
Cell Discovery Port (CELL_DISCOVERY_ADDRESS)	Not applicable	7277	
CELL_MULTICAST_DISCOVERY_ADDRESS	Not applicable	7272	
NODE_MULTICAST_IPV6_DISCOVERY_ADDRESS	5001	5001	

When you federate an application server node into a deployment manager cell, the deployment manager creates a node agent server on the application server node. If you do not specify values for the port assignments, the node agent uses the default values.

Table 4. Default port definitions for the Version 6.0.x node agent server process

Port Name	Default Value	File
Name Service or RMI Connector Port (BOOTSTRAP_ADDRESS)	2809	serverindex.xml
SOAP Port (SOAP_CONNECTOR_ADDRESS)	8878	
Data Replication Service Client Port (DRS_CLIENT_ADDRESS)	7888	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9901	
Common Secure Interoperability Version 2 (CSIV2) Transport Port (CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS)	9101	
CSIV2 Transport Port (CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS)	9201	
Object Request Broker (ORB) Listener Port (ORB_LISTENER_ADDRESS)	9900	
Node Discovery Port (NODE_DISCOVERY_ADDRESS)	7272	
Node Multicast Discovery Port (NODE_MULTICAST_DISCOVERY_ADDRESS)	5000	

Version 5.x port numbers

Table 5. Port definitions for WebSphere Application Server Version 5.x

Port Name	Default Value		Files
	WebSphere Application Server	Network Deployment	
HTTP_TRANSPORT	9080	Not applicable	server.xml and virtualhosts.xml
HTTPS_TRANSPORT	9443	Not applicable	
HTTP_TRANSPORT_ADMIN	9090	9090	
HTTPS_TRANSPORT_ADMIN	9043	9043	
JMSSERVER_SECURITY_PORT	5557	Not applicable	server.xml
JMSSERVER_QUEUED_ADDRESS	5558	Not applicable	serverindex.xml
JMSSERVER_DIRECT_ADDRESS	5559	Not applicable	
BOOTSTRAP_ADDRESS	2809	9809	
SOAP_CONNECTOR_ADDRESS	8880	8879	
DRS_CLIENT_ADDRESS	7873	7989	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	0	9401	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	0	9403	
CSIV2_SSL_MULTIAUTH_LISTENER_ADDRESS	0	9402	
IBM HTTP Server Port	80	Not applicable	virtualhosts.xml, plugin-cfg.xml, and *web_server_root*/conf/httpd.conf
IBM HTTPS Server Administration Port	2001	Not applicable	Not applicable
CELL_DISCOVERY_ADDRESS	Not applicable	7277	serverindex.xml
ORB_LISTENER_ADDRESS	9100	9100	
CELL_MULTICAST_DISCOVERY_ADDRESS	Not applicable	7272	

When you federate an application server node into a deployment manager cell, the deployment manager instantiates the node agent server process on the application server node. The node agent server uses these port assignments by default.

Table 6. Port definitions for the Version 5.x node agent server process

Port Name	Default Value	File
BOOTSTRAP_ADDRESS	2809	serverindex.xml
ORB_LISTENER_ADDRESS	9900	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9901	
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9101	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9201	
NODE_DISCOVERY_ADDRESS	7272	
NODE_MULTICAST_DISCOVERY_ADDRESS	5000	
DRS_CLIENT_ADDRESS	7888	
SOAP_CONNECTOR_ADDRESS	8878	

Chapter 2. Communicating with Web servers

The WebSphere Application Server works with a Web server to route requests for dynamic content, such as servlets, from Web applications. The Web servers are necessary for directing traffic from browsers to the applications that run in WebSphere Application Server. The Web server plug-in uses the XML configuration file to determine whether a request is from the Web server or the Application Server.

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

  If you want to use an IBM HTTP Server, see the *Installing your application serving environment* PDF. Otherwise, see the installation information provided with your Web server.
- Ensure that your Web server is configured to perform operations required by Web applications, such as GET and POST. Typically, this involves setting a directive in the Web server configuration file (such as the httpd.conf file for an IBM HTTP Server). Refer to the Web server documentation for instructions. If an operation is not enabled when a servlet or JSP file requiring the operation is accessed, an error message displays, such as this one from the IBM HTTP Server:
- Make sure that the i5/OS Web server plug-ins are installed. The i5/OS Web server plug-ins are normally installed as a component of the WebSphere Application Server product during product installation.

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

1. The appropriate plug-in file is installed to your Web server and the script configureWeb_server_name created is run to create and configure the Web server definition in the WebSphere configuration repository.

   During the plug-in installation process, an unmanaged node is created if the Web server is on a different computer from the Application Server. An unmanaged node is a node that does not have a WebSphere node agent running on it. Using unmanaged nodes, WebSphere Application Server can represent servers that are not application servers within its configuration topology. This representation enables connection information between those servers and application servers to be maintained. The *Using the administrative clients* PDF describes how to create a node. See the Plug-in Installation Roadmap for additional information.
2. A Web server definition is created.

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

   a. Select the node that was created in the preceding step, and in the Server name field, enter the local name of the Web server for which you are creating a Web server definition.
   b. Use the wizard to complete the Web server definition.
3. An application or modules are mapped to a Web server. If an application that you want to use with this Web server is already installed, the application is automatically mapped to the Web server. If the application is not installed, select this Web server during the Map modules to servers step of the application installation process.
4. The master repository is updated and saved.

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

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

The configuration is complete. To activate the configuration, stop and restart the Web server. If you encounter problems restarting your Web server, check the http_plugin.log file for information on what

portion of the plugin-cfg.xml file contains an error. The log file states the line number on which the error occurred along with other details that might help you diagnose why the Web server did not start. You can then use the administrative console to update the plugin-cfg.xml file.

If applications are infrequently installed or uninstalled, which is usually the situation in a production environment, or if you can tolerate the performance impact of generating and distributing the plug-in configuration file each time any of the previously listed actions occur, you should consider enabling this service.

If you are making a series of simultaneous changes, like installing numerous applications, you might want the configuration service disabled until after you make the last change. The Web server plug-in configuration service is enabled by default. To disable this service, in the administrative console click elect **Servers > Application Servers >** *server_name* **> Administration Services >Web server plug-in configuration service** and then unselect the Enable automated Web server configuration processing option.

Tip: If your installation uses a firewall, make sure you configure the Web server plug-in to use a port that has been opened. (See your security administrator for information on how to obtain an open port.)

Installing IBM HTTP Server

IBM HTTP Server for iSeries. The IBM HTTP Server Version 6 product is included in the media package when you order any of the WebSphere Application Server V6.x for i5/OS products. It is not a required product. To install IBM HTTP Server V6 on a non-iSeries server, use the following instructions.

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

To use a Web server other than IBM HTTP Server Version 6.x, install and configure the Web server before or after installing the WebSphere Application Server product, but before installing the Web server plug-ins for WebSphere Application Server.

The Plug-ins installation wizard configures supported Web servers. You can also manually configure supported Web servers for WebSphere Application Server, Version 6, as described in “Editing Web server configuration files” on page 72.

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

After installing the Web server and the Application Server, install the appropriate Web server plug-in for a supported, installed Web server. No further configuration is required for most Web servers.

This topic describes installing IBM HTTP Server.

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

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

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

   Separate installation procedures for the WebSphere Application Server product, the IBM HTTP Server product, and the Web server plug-ins let you install only what you need on a particular machine.

4. Click **Next** to display the License agreement panel.
5. Accept the license agreement and click **Next** to display the installation root directory panel.
6. Specify the root directory information and click **Next** to display the feature type selection panel.

   The panel lets you bypass features selection by accepting typical features. Selecting Custom lets you select features in the Features selection panel.
7. Click **Custom** to select features and click **Next** to display the Features selection panel.
8. Review the confirmation panel to verify your selections. Click **Back** to change any of your specifications. Click **Next** to begin installing IBM HTTP Server.
9. Review the confirmation panel to verify your selections. Click **Back** to change any of your specifications. Click **Next** to begin installing IBM HTTP Server.

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

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

You can install the IBM HTTP Server product.

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

Installing Web server plug-ins

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

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

The Plug-ins installation Wizard is not used for Web servers on i5/OS. Instead, install the Web server plug-ins component during the WebSphere Application Server installation. Additional details about plug-in scenarios are described later in this topic.

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

The Plug-ins installation wizard installs the plug-in module, configures the Web server for communicating with the application server, and creates a Web server configuration definition in the application server, if possible.

Some topologies, such as the Web server on one machine and the application server on another machine, prevent the Plug-ins installation wizard from creating the Web server definition in the application server

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

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

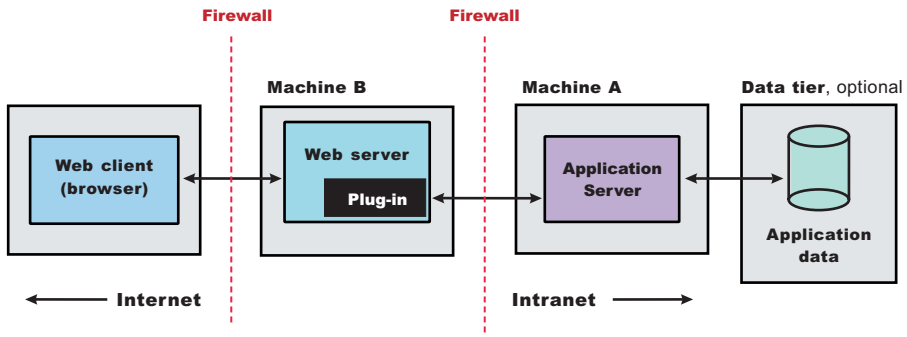
The Plug-ins installation wizard installs required files and configures the Web server and the application server to allow communication between the servers.

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

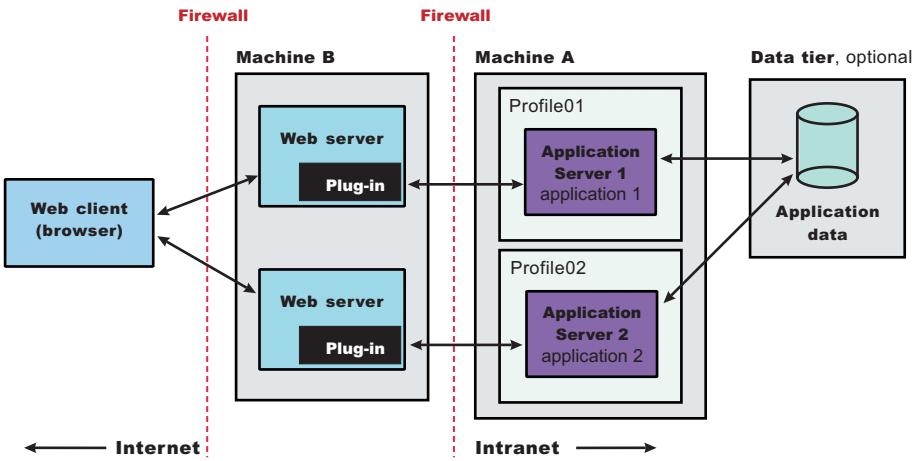
The following descriptions use separate machines for installable components of the working system, but the concepts also apply to separate logical partitions on iSeries systems.

When multiple profiles exist, the plug-ins installer configures only the default profile. See “Plug-ins configuration” on page 23 for a description of the flow of logic that determines how the installer selects the profile to configure.

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

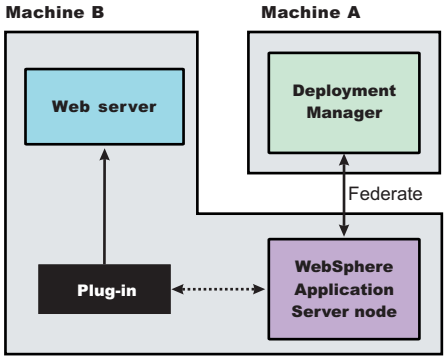


- **Scenario 2: Remote** Multiple stand-alone application servers are on one machine, and each application server has a dedicated Web server on a separate machine or logical partition.



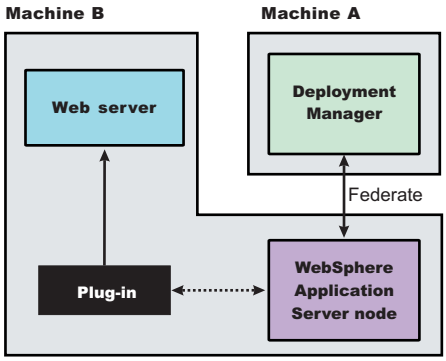
- **Scenario 3: Local Application Server profile** The application server and the Web server are on a single machine or logical partition.

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



- **Scenario 4: Local custom profile** A managed node and the Web server are on the same machine or logical partition.

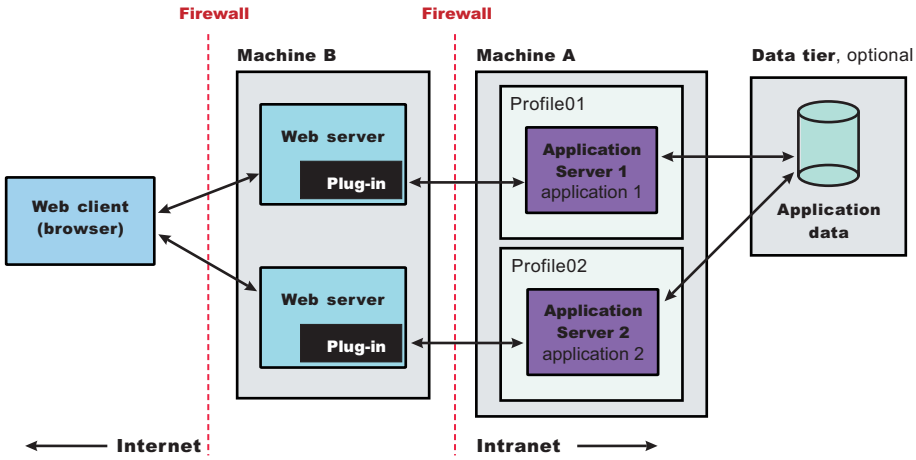
  A local distributed installation includes the Web server plug-in, the Web server, and the managed custom node on the same machine:



- **Scenario 5: Local deployment manager profile** A deployment manager node and the Web server are on a single machine or logical partition.

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

- **Scenario 6: Non-default profile** Creating a Web server definition for a profile that is not the default profile.



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

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

See “Web server configuration” on page 30 for more information about the files involved in configuring a Web server.

See “Editing Web server configuration files” on page 72 for information about how the Plug-ins installation wizard configures supported Web servers.

Selecting a Web server topology diagram and roadmap

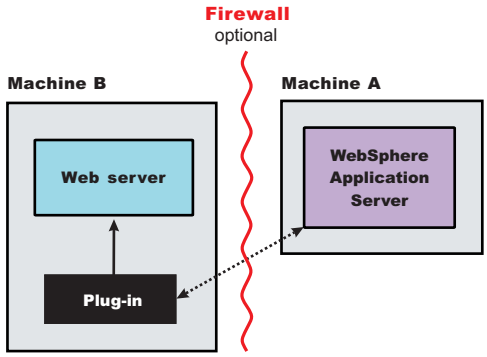
Install and configure Web server plug-ins for WebSphere Application Server to allow the application server to communicate with the Web server.

The primary production configuration for a Web server is an application server on one machine and a Web server on a separate machine. This configuration is referred to as a *remote* configuration. Contrast the remote configuration to the local configuration, where the application server and the Web server are on the same machine.

- **Set up a remote Web server installation for a Standalone node.**

  The remote Web server configuration is recommended for production environments.

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



Remote installation scenario

Table 7. Installation and configuration

Step	Machine	Task
1	A	Install your WebSphere Application Server product.
2	A	Create an application server profile.
3	B	If you plan to run IBM HTTP Server on iSeries, it is already installed as product 5722DG1. You can also use a Domino Web server on iSeries. Refer to the Domino documentation for installation instructions. For either scenario, you must install the Web server Plug-ins component of the WebSphere Application Server product.
4	B	Run the **manageprofiles** Qshell command to create an http profile. The http profiles are V6.x equivalents of 5.0 and 5.1 remote instances. For example, run this command from Qshell: `app_server_root/bin/manageprofiles` `  -create` `  -profileName myHttpProfile` `  -templatePath http` The *myHttpProfile* variable is the name of the profile.

Table 7. Installation and configuration (continued)

Step	Machine	Task
5	B	Configure the IBM HTTP Server with your http profile myHttpProfile. If your Web server's name is MyWebServer., an i5/OS qshell script called *configureIHS_MyWebServer* is created in the myHttpProfile_profile_root/config/IHS_myWebServer directory on machine B. For the default WebSphere Application Server install, the*myHttpProfile_profile_root* of the profile myHttpProfile is /QIBM/UserData/WebSphere/AppServer/V61/Base/profiles/myHttpProfile. **Note:** In the remainder of this example, webServerName refers to IHS_myWebServer. If you choose to configure a DOMINO Web server as listed below, then webServerName refers to DOMSRV01. The following steps apply to **DOMINO Web servers only**: 1. Run the *configureOs400WebServerDefinition* script on the http profile myHttpProfile. For example: `configureOs400WebServerDefinition -profileName myHttpProfile -webserver.name DOMSRV01 -webserver.type DOMINO -webserver.port 80` 2. Using the WRKDOMSVR command to update the notes.ini file of your Domino server, insert the following directive: WebSphereInit=myHttpProfile_profile_root/config/DOMSRV01/plugin-cfg.xml 3. From the Lotus Notes client connected to the Domino server, click the internet protocols tab and then click the HTTP tab. Under DSAPI filter names add the following: /QSYS.LIB/ *product_lib*.LIB/LIBDOMINO.SRVPGM 4. Save your changes
6	A	Copy the *configurewebServerName* script from Machine B to Machine A. The script is found in the myHttpProfile_profile_root/config/webServerName directory described above
7	A	Place the file you copied from the previous step into the profile_root/bin directory on Machine A, where *profile_root* is the directory where your application server profile is located.
8	A	Start the application server and then run the script that you copied in the previous step. For example, run these commands from Qshell: `app_server_root/bin/startServer -profileName myProfile` `cd profile_root/bin` `./configurewebServerName [wasAdminUserId] [wasAdminPassword]` **Note:** *wasAdminUserId* and *wasAdminPassword* are optional and only needed when the application server of *myProfile* is running in secure mode.
9	A	If you use IBM HTTP Server on iSeries, verify that the application server is running. Open the administrative console (ISC) and do the following: 1. Expand Servers > Web servers 2. Select your Web server, in this case it is *IHS_MyWebServer*, then click Remote web server management 3. Enter the user ID and password used to authenticate to Machine B. The authorities required by this profile are the same as that required to access the HTTP administration GUI. For details, see User profiles and required authorities for HTTP Server in the iSeries Information Center. 4. Save your configuration.
10	A	Configure a virtual host alias for the Web server machine(B) and Web server port of MyWebServer. .

Table 7. Installation and configuration (continued)

Step	Machine	Task
11	A	Stop and restart your application server.
12	A	In the administrative console (ISC) do the following: 1. Select *webServerName* and click Generate Plug-in to generate the plugin-cfg.xml file. 2. Select *webServerName* and click Propagate Plug-in to propagate the plugin-cfg.xml file to machine B.
13	B	If you use IBM HTTP Server on iSeries, start the Web server. Open the administrative console (ISC) and do the following: 1. Expand Servers > Web servers. 2. Select your Web server, in this case it is *IHS_MyWebServer*, then click Start. If you use Domino HTTP Server on iSeries, start the Web server from a CL command line: 1. Run the Work with Domino Servers (WRKDOMSVR) command. 2. Specify option 1 next to your Domino server. 3. Press Enter.
14	B	Run the snoop servlet. Access the following URL in your browser: `http://host_name_of_machine_B/snoop` If you get an error, retrace your steps. Add a virtual host to Machine A before restarting the application server on Machine A.

Regeneration of the plugin-cfg.xml file

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

However, if the service is disabled or if you want to force regeneration, use the administrative console or the GenPluginCfg script. In the administrative console, perform these steps:

1. Expand **Servers > Web servers** .
2. Select the Web server for which you want to regenerate the plugin-cfg.xml file.
3. Click **Generate Plugin**.

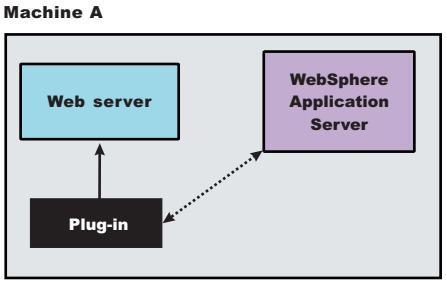
Manual propagation of the plugin-cfg.xml file (Required for DOMINO Web servers)

The plugin-cfg.xml file can be propagated manually as follows: Copy the plugin-cfg.xml file from the application server machine to the myHttpProfile_profile_root/config/IHS_MyWebServer directory on the Web server machine B. The plugin-cfg.xml file is generated in the directory named profile_root/config/cells/cell_name/nodes/host_name_of_machine_B-node/servers/IHS_myWebServer on the application server machine A.

- **Set up a local Web server configuration for a Standalone node.**

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

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



Local installation scenario

Table 8. Installation and configuration

Step	Machine	Task
1	A	Install your WebSphere Application Server product.
2	A	Create an application server profile.
3	A	IBM HTTP Server on iSeries is already installed as product 5722DG1. Alternatively, you can also run Domino Web Server on iSeries. Refer to the Domino documentation for installation instructions.
4	A	If using IBM HTTP Server on iSeries, verify that the plug-in component of WebSphere Application Server is installed. From a CL command line, run this command: `wrklnk '/qsys.lib/product_lib.lib/qsvtap20.srvpgm'` If the service program exists, the plug-in component is installed. If the object is not found, select the Web server plug-ins component during the installation of the WebSphere Application Server product.
5	A	Configure the IBM HTTP Server with your WebSphere Application Server profile. The following steps apply to DOMINO Web servers only. For these steps, assume your Web server's name is *MyWebServer*. 1. Run the *configureOs400WebServerDefinition* script on the application server profile. For example: `configureOs400WebServerDefinition -profileName myAppServerProfile -webserver.name DOMSRV01 -webserver.type DOMINO -webserver.port 80` 2. Configure a virtual host alias for the Web server machine and Web server port of DOMSRV01. 3. Using the WRKDOMSVR command to update the notes.ini file of your Domino server, insert the following directive: `WebSphereInit=profile_root/config/cells/cell_name/nodes/node_name/servers/DOMSRV01/plugin-cfg.xml` 4. From the Lotus Notes client connected to the Domino server, click the Internet protocols tab, then click the HTTP tab. Under DSAPI filter names add the following: /QSYS.LIB/ *product_lib*.LIB/LIBDOMINO.SRVPGM 5. Save your changes
6	A	Stop and restart the application server.

Table 8. Installation and configuration (continued)

Step	Machine	Task
7	A	If you use IBM HTTP Server on iSeries, open the administrative console (ISC) do the following: 1. Expand Servers > Web servers 2. Select your Web server, in this case it is *IHS_MyWebServer*, then click Remote web server management. 3. Enter the user ID and password used to authenticate to Machine A. The authorities required by this profile are the same as that required to access the HTTP administration GUI. For details, see User profiles and required authorities for HTTP Server in the iSeries Information Center. 4. Save your changes.
8	A	In the administrative console (ISC) do the following: 1. Expand Servers > Web servers. 2. If you use IBM HTTP Server on iSeries, Select *IHS_MyWebServer* and click Generate Plug-in to generate the plugin-cfg.xml file. 3. If you use Domino HTTP Server on iSeries, Select DOMSRV01 and click Generate Plug-in to generate the plugin-cfg.xml file.
9	A	If you use IBM HTTP Server on iSeries, start the Web server. Open the administrative console (ISC) and do the following: 1. Expand Servers > Web servers. 2. Select your Web server, in this case it is *IHS_MyWebServer*, then click Start. If you use Domino HTTP Server on iSeries, start the Web server from a CL command line: 1. Run the Work with Domino Servers (WRKDOMSVR) command. 2. Specify option 1 next to your Domino server. 3. Press Enter.
10	A	Run the snoop servlet. Access the following URL in your browser: `http://host_name_of_machine_A/snoop` If you get an error, retrace your steps. Add a *virtual host* to Machine A before restarting the application server on Machine A.

Regeneration of the plugin-cfg.xml file

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

However, if the service is disabled or you want to force regeneration, use the administrative console or the GenPluginCfg script. In the administrative console, perform these steps:

1. Expand **Servers > Web servers**.
2. Select the Web server for which you want to regenerate the plugin-cfg.xml file.
3. Click **Generate Plug-in**.

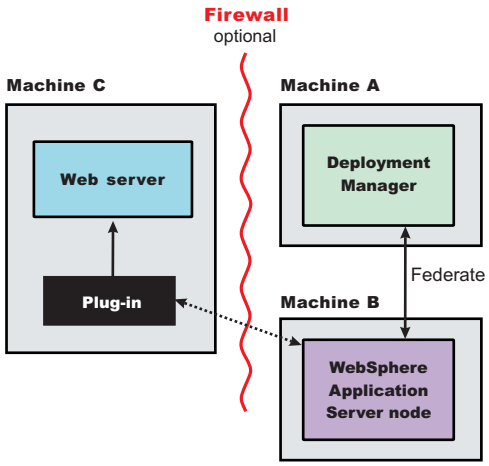
Propagation of the plugin-cfg.xml file

The local file does not require propagation.

- **Set up a remote Web server configuration for a Managed node.**

  The remote Web server configuration is recommended for production environments.

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



Remote distributed installation scenario

Table 9. Installation and configuration

Step	Machine	Task
1	A	Install WebSphere Application Server Network Deployment.
2	A	Create a deployment manager profile or use the one created during installation.
3	A	Start the deployment manager. From Qshell, run this command: `app_server_root/bin/startManager` `-profileName name_of_dmgr_profile`
4	B	Install WebSphere Application Server Network Deployment.
5	B	Create an application server profile or use the one created during installation.
6	B	Federate the node. From Qshell, run this command: `app_server_root/bin/addNode` `dmgrHost` `SOAP_port` `-profileName appProfile` `-includeapps` Variable descriptions: • The *dmgrHost* variable is the host name of the machine where your deployment manager profile exists. • The *SOAP_port* variable is the SOAP port of the dmgr profile. The default SOAP port is 8879. To determine which port your dmgr profile uses, run the following command: `app_server_root/bin/dspwasinst` `-profileName name_of_dmgrProfile` • The *appProfile* variable is the name of the application server profile that you want to federate. Map the application modules to servers after this step.
7	C	IBM HTTP Server on iSeries is already installed as product 5722DG1. Alternatively, you can also run Domino Web Server on iSeries. Refer to the Domino documentation for installation instructions. For either scenario, you must install the Web server Plug-ins component of the WebSphere Application Server product.

Table 9. Installation and configuration (continued)

Step	Machine	Task
8	C	Run the **manageprofiles** Qshell command to create an http profile. The http profiles are V6.x equivalents of 5.0 and 5.1 remote instances. For example, run this command from Qshell: `app_server_root/bin/manageprofiles` `  -create` `  -profileName myHttpProfile` `  -templatePath http` The *myHttpProfile* variable is the name of the profile.
9	C	Configure the IBM HTTP Server with your http profile myHttpProfile. Assuming your Web server's name is MyWebServer. an i5/OS qshell script called *configureIHS_MyWebServer* will be created in the myHttpProfile_profile_root/config/IHS_MyWebServer directory on machine C. For the default WebSphere Application Server install, the *myHttpProfile_profile_root* of the profile myHttpProfile is /QIBM/UserData/WebSphere/AppServer/V61/Base/profiles/myHttpProfile. **Note:** In the remainder of this example, webServerName refers to IHS_myWebServer. If you choose to configure a DOMINO Web server as listed below, then webServerName refers to DOMSRV01. The following steps apply to **DOMINO Web servers only**: 1. Run the *configureOs400WebServerDefinition* script on the http profile myHttpProfile. For example: `configureOs400WebServerDefinition -profileName myHttpProfile` `  -webserver.name DOMSRV01 -webserver.type DOMINO -webserver.port 80` 2. Using the WRKDOMSVR command to update the notes.ini file of your Domino server, insert the following directive: WebSphereInit=myHttpProfile_profile_root/config/DOMSRV01/plugin-cfg.xml 3. From the Lotus Notes client connected to the Domino server, click the Internet protocols tab, then click the HTTP tab. Under DSAPI filter names add the following: /QSYS.LIB/ *product_lib*.LIB/LIBDOMINO.SRVPGM 4. Save your changes.
10	C	Copy the configure*webserverName* script to Machine A. The script is found in the myHttpProfile_profile_root/config/webServerName directory described above.
11	A	Place the file you copied from the previous step into the *profile_root*/bin directory on Machine A, where profile_root is the directory where your deployment manager profile is located.
12	A/B	Start the node agent and the deployment manager if they are not already running, then run the script that you copied in the previous step. For example, run these commands from Qshell: **Machine A:** `app_server_root/bin/startManager` `-profileName name_of_dmgr_profile` `cd profile_root/bin` `./configurewebServerName [wasAdminUserId] [wasAdminPassword]` **Note:** *wasAdminUserId* and *wasAdminPassword* are optional and only needed when the deployment manager of *name_of_dmgr_profile* is running in secure mode. **Machine B:** `app_server_root/bin/startNode` `-profileName appProfile`

Table 9. Installation and configuration (continued)

Step	Machine	Task
13	A	Use the administrative console of the deployment manager on Machine A to start the application server on Machine B. Wait for synchronization to occur and save the new configuration.
14	A	If you use IBM HTTP Server on iSeries, verify that the deployment manager is running. Open the administrative console (ISC) and do the following: 1. 1. Expand Servers > Web servers 2. 2. Select your Web server, in this case it is IHS_MyWebServer, then click Remote web server management. . 3. 3. Enter the user ID and password used to authenticate to Machine C. The authorities required by this profile are the same as that required to access the HTTP administration GUI. For details, see User profiles and required authorities for HTTP Server in the iSeries information center. 4. 4. Save your configuration.
15	A	Configure a virtual host alias for the Web server machine(C) and Web server port of MyWebServer.
16	A	In the administrative console (ISC) do the following: 1. Select webServerName and click Generate Plug-in to generate the plugin-cfg.xml file. 2. Select webServerName and click Propagate Plug-in to propagate the plugin-cfg.xml file to machine C.
17	A	Stop and restart the federated application server.
18	A/C	If you use IBM HTTP Server on iSeries, start the Web server. Open the administrative console (ISC) on machine A and do the following: 1. Expand Servers > Web servers. 2. Select your Web server, in this case it is IHS_MyWebServer, then click Start. If you use Domino HTTP Server on iSeries, start the Web server, on machine C, from a CL command line: 1. Run the Work with Domino Servers (WRKDOMSVR) command. 2. Specify option 1 next to your Domino server. 3. 3. Press Enter.
19	C	Run the snoop servlet. Access the following URL in your browser: `http://host_name_of_machine_C/snoop` If you get an error, retrace your steps.

Regeneration of the plugin-cfg.xml file

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

However, if the service is disabled or you want to force regeneration, use the administrative console or the **GenPluginCfg** script. In the administrative console, perform these steps:

1. Expand **Servers > Web servers**.
2. Select the Web server for which you want to regenerate the plugin-cfg.xml file.
3. Click **Generate Plug-in**.

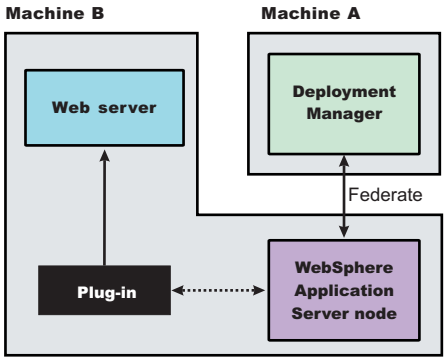
Manual propagation of the plugin-cfg.xml file (Required for DOMINO Web servers)

The plugin-cfg.xml file can be propagated manually as follows: Copy the plugin-cfg.xml file from the application server machine to the myHttpProfile_profile_root/config/IHS_MyWebServer directory on the Web server machine C. The plugin-cfg.xml file is generated in the directory named profile_root/config/cells/cell_name/nodes/node_name/servers/IHS_myWebServer on the application server machine B.

- **Set up a local Web server configuration for a Managed node.**

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

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



Local distributed installation scenario

Table 10. Installation and configuration

Step	Machine	Task
1	A	Install WebSphere Application Server Network Deployment.
2	A	Create a deployment manager profile or use the one created during installation.
3	A	Start the deployment manager. From Qshell, run this command: *app_server_root*/bin/startManager -profileName *name_of_dmgr_profile* Alternatively, issue the following commands: cd *profile_root*/bin startManager
4	B	Install WebSphere Application Server Network Deployment.
5	B	Create an application server profile or use the one created during the installation. Assume the profile name is appProfile.
6	B	Federate the node. From Qshell, run the following commands: *app_server_root*/bin/addNode *dmgrHost* *SOAP_port* -profileName *appProfile* -includeapps Variable descriptions: • The *dmgrHost* variable is the host name of the machine where your deployment manager profile exists. • The *SOAP_port* variable is the SOAP port of the dmgr profile. The default SOAP port is 8879. To determine which port your dmgr profile uses, run the following command: *app_server_root*/bin/dspwasinst -profileName *name_of_dmgrProfile* • The *appProfile* variable is the name of the application server profile that you want to federate. Map the application modules to servers after this step.

Table 10. Installation and configuration (continued)

Step	Machine	Task
7	B	IBM HTTP Server on iSeries is already installed as product 5722DG1. Alternatively, you can also run Domino Web Server on iSeries. Refer to the Domino documentation for installation instructions.
8	B	Configure the IBM HTTP Server with your application server profile appProfile. **Note:** In the remainder of this example, webServerName refers to IHS_MyWebServer. If you choose to configure a DOMINO Web server as listed below, then webServerName refers to DOMSRV01. Also in this example, assume your Web server's name is MyWebServer.
9	A/B	**(Domino only)** The following steps apply to the DOMINO Web server on machine B: 1. Run the *configureOs400WebServerDefinition* script on the deployment manager profile. For example: `configureOs400WebServerDefinition -profileName name_of_dmgr_profile -webserver.name DOMSRV01 -webserver.type DOMINO -webserver.node name_of_federated_node -webserver.port 80.` In this case, *name_of_federated_node* is *name_of_machine_B_appProfile*. 2. Using the administrative console (ISC) on machine A, configure a virtual host alias for the Web server machine and Web server port of DOMSRV01. 3. On machine B, Use the WRKDOMSVR command to update the notes.ini file of your Domino server. Insert the following directive: WebSphereInit=appProfile_root/config/cells/cell_name/nodes/node_name/servers/DOMSRV01/plugin-cfg.xml. *appProfile* is the application server profile name on machine B. 4. On machine B, from the Lotus Notes client connected to the Domino server, click the Internet protocols tab, then click the HTTP tab. Under DSAPI filter names add the following: /QSYS.LIB/ *product_lib*.LIB/LIBDOMINO.SRVPGM 5. Save your changes.
10	A	Use the administrative console of the deployment manager on Machine A to start the application server on Machine B.
11	A	If you use IBM HTTP Server on iSeries, update the user ID and password of your Web server configuration. This step enables your deployment manager to perform remote operations on the Web server: 1. Open the administrative console for your deployment manager profile on Machine A. 2. Expand **Servers > Web servers** 3. Select your Web server, in this case it is IHS_MyWebServer, then click **Remote web server management** 4. Enter the user ID and password used to authenticate to Machine B. The authorities required by this profile are the same as that required to access the HTTP administration GUI. For details, see User profiles and required authorities for HTTP Server in the iSeries information center 5. Save your configuration.
12	A	In the administrative console (ISC), configure a virtual host alias for the Web server machine(B) and Web server port of MyWebServer.
13	A	In the administrative console (ISC) do the following: 1. Select *webServerName* and click **Generate Plug-in** to generate the plugin-cfg.xml file. 2. Select *webServerName* and click **Propagate Plug-in** to propagate the plugin-cfg.xml file.
14	A	In the administrative console (ISC), stop and restart the federated application server.

Table 10. Installation and configuration (continued)

Step	Machine	Task
15	A/B	If you use IBM HTTP Server on iSeries, start the Web server. Open the administrative console (ISC) on machine A and do the following: 1. Expand **Servers > Web servers** 2. 2. Select your Web server, in this case it is IHS_MyWebServer, then click Start. If you use Domino HTTP Server on iSeries, start the Web server, on machine B, from a CL command line: 1. Run the Work with Domino Servers (WRKDOMSVR) command. . 2. Specify option 1 next to your Domino server. 3. Press Enter. .
16	B	Run the snoop servlet. Access the following URL in your browser: `http://host_name_of_machine_B/snoop` If you get an error, retrace your steps.

Regeneration of the plugin-cfg.xml file

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

However, if the service is disabled or you want to force regeneration, use the administrative console or the GenPluginCfg script. In the administrative console, perform these steps:

1. Expand **Servers > Web servers**.
2. Select the Web server for which you want to regenerate the plugin-cfg.xml file.
3. Click **Generate Plug-in**.

The plugin-cfg.xml file is generated at the location *profile_root*/config/cells/*cell_name*/nodes/*node_name*/servers/*webServerName* directory, when the Web server definition is created. In the above table, the proile_root is the root of the deployment manager profile on machine A.

Regenerate the plugin-cfg.xml file in the Web server definition in the application server whenever the configuration changes. The Web server has immediate access to the file whenever it is regenerated.

Propagation of the plugin-cfg.xml file

Node synchronization is used to propagate the plugin-cfg.xml file from Machine A to Machine B.

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

See “Web server configuration” on page 30 for more information about the files involved in configuring a Web server.

See “Editing Web server configuration files” on page 72 for information about how the Plug-ins installation wizard configures supported Web servers.

See “Installing Web server plug-ins” on page 9 for information about other installation scenarios for installing Web server plug-ins.

Plug-ins configuration

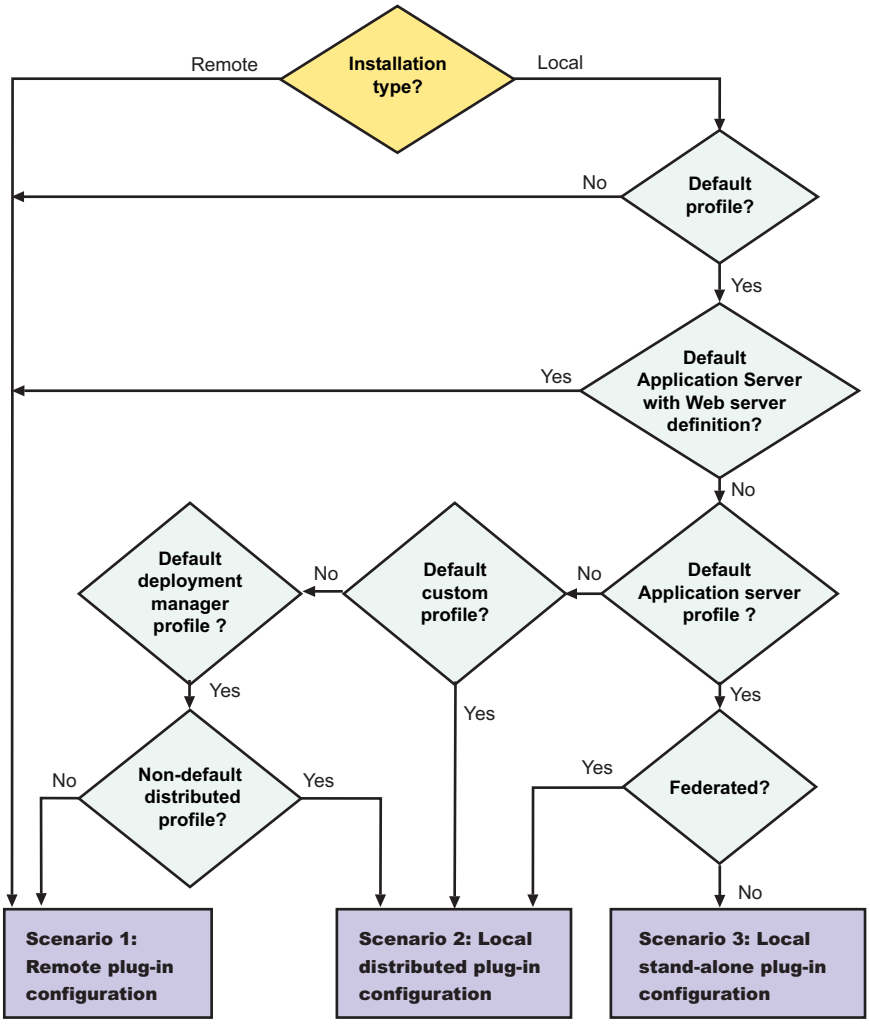
The Plug-ins installation wizard installs a binary plug-in module and a plug-in configuration file for the Web server. The wizard then configures the supported Web server for the Application Server and creates a Web server definition in the configuration of the application server. This overview shows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

This topic describes the three ways that the Plug-ins installation wizard configures a Web server to locate the plugin-cfg.xml file, which is the plug-in configuration file.

Configuration flows for the Network Deployment product

The Plug-ins installation wizard resolves all configurations of Web server and WebSphere Application Server to three scenarios: a remote application server, local distributed application Server, and local stand-alone Application Server. The logic implemented in determining which scenario applies to a configuration is shown in the following diagram.

Web server plug-ins for WebSphere Application Server



Legend:

Default application server with Web server definition?

If the default profile is an application server with an existing Web server definition, then the installation is considered a remote installation. You cannot have more than one Web server definition in a stand-alone application server.

Use the same name for the Web server to configure a new Web server to use the existing Web server definition.

Use a different Web server name to create a script that creates a new Web server definition in a federated application server profile.

Default profile?
If the product is installed but the Profile Management tool has not yet created a profile, the scenario is considered to be a remote installation. When multiple profiles exist, the plug-ins installer configures only the default profile.

Default *profile_type***?**
The Plug-ins installation wizard can configure only one profile at a time. The wizard always works with the default profile. These three paths show how processing varies for different types of profiles.

Federated?
If the application server node is federated, the Plug-ins installation wizard configures the Web server definition on the managed node. This has advantages. Suppose the Web server and the managed node are on a separate machine. The plugin-cfg.xml file is automatically propagated to the remote node during node synchronization because the Web server definition is part of the node configuration.

Installation type?
The installation type is either remote or local.

Non-default distributed profile?
If the deployment manager has a federated custom node (custom profile), the Plug-ins installation wizard configures the Web server definition on the managed node. This has advantages. Suppose the Web server and the managed node are on a separate machine. The plugin-cfg.xml file is automatically propagated to the remote node during node synchronization because the Web server definition is part of the node configuration.

If a federated custom profile is not found, the Plug-ins installation wizard looks for and configures the first federated application server node (application server profile) that it finds. So, the logic is:

1. Look for a federated managed (custom) profile and configure the first one found.
2. If no federated managed profile is found, look for a federated application server profile and configure the first one found.

Scenario 1. Remote plug-in configuration

The Plug-ins installation wizard does not automatically create a Web server definition within the default distributed profile on a remote machine. The wizard creates the configure*web_server_name* script instead.

The Plug-ins installation wizard configures the Web server to use the plugin-cfg.xml file that will be maintained on the Web server machine in the *plugins_root*/config/*web_server_name* directory. This file requires periodic propagation. Propagation is copying the current plugin-cfg.xml file from the Application Server machine to replace the *plugins_root*/config/*web_server_name*/plugin-cfg.xml file.

After installing the binary plug-in for the local Web server, you do not have to run the script before you can start the application server and the Web server. However, you do not have the benefits of a Web server definition in the application server node until you run the script.

Four configurations qualify for the remote application server scenario:

Profile type	Federation status	Creation of Web server definition?	Web server already defined in Application Server configuration?
Any profile anywhere if you select a remote installation type in the Plug-ins installation wizard	N/A	By script	N/A
No default profile detected	N/A	By script	N/A
Default unfederated stand-alone application server profile with an existing Web server definition	Not federated	By script	Yes

Profile type	Federation status	Creation of Web server definition?	Web server already defined in Application Server configuration?
Default deployment manager profile with no managed nodes	N/A	By script	N/A

Testing the application server without a Web server definition: The following overview shows the procedure for verifying the temporary *plugins_root*/config/*web_server_name*/plugin-cfg.xml file.

The Web server communicates with the remote Application Server using the temporary plugin-cfg.xml file.

If the application server has an HTTP Transport port assignment other than 9080, the test is not successful. Continue to the next section to create the Web server definition on the application server and complete your test of the configuration.

1. Start the Web server with the proper procedure for your Web server.

   For example, start the IBM HTTP Server from a command line:
   - ./*IHS_root*/bin/apachectl start
2. Start the application server on the remote machine.

   Change directories to the *profile_root*/bin directory and run the startServer command:
   - `./profile_root/bin/startServer server1`
3. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the application server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the Web server plug-in.
4. Verify that both Web addresses display the Snoop Servlet - Request/Client Information page.

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

1. Start the deployment manager if you are configuring the deployment manager or a managed node.
2. Federate a remote application server node or custom node now if you are planning to federate the node at some point. If a Web server definition already exists when you federate a node, the definition is lost.
3. Create the Web server definition in the application server. You have two options for a managed node. Use the script option for a deployment manager node without managed nodes.
   - Use the administrative console of the deployment manger to create a Web server definition for a managed node. Click **Servers > Web servers > New** and use the Create new Web server entry wizard to create the Web server definition.
   - Run the script to manually create the Web server definition within the configuration of the Application Server node:
     a. Copy the script from the *plugins_root*/bin directory to the remote *app_server_root*/bin directory.
     b. Open a command window and run the script:
        – ./configure*web_server_name*

     If you have enabled security or changed the default Java Management Extensions (JMX) connector type, edit the script and include the appropriate parameters.
4. Open the administrative console of the deployment manager if the node is federated. Wait for node synchronization to occur on the managed node and save the changed configuration that includes the new Web server definition. If the remote node is not federated, open the administrative console of the application server and save the changed configuration.

5. Copy the current plug-in configuration file, plugin-cfg.xml, in the *profile_root*/config/cells/*cell_name*/nodes/*web_server_name*_node/servers/*web_server_name* directory. Paste the file on the Web server machine to replace the temporary *plugins_root*/config/*web_server_name*/plugin-cfg.xml file. The IBM HTTP Server supports automatic propagation. Other Web servers require manual propagation.
6. Start the Web server with the proper procedure for your Web server.
7. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the Application Server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the Web server plug-in.
8. Verify that both Web addresses display the Snoop Servlet - Request/Client Information page.

Scenario 2. Local distributed plug-in configuration

The Plug-ins installation wizard does not automatically create a Web server definition within a federated application server profile. The wizard creates the configure*web_server_name* script instead in the *plugins_root*/bin directory.

The Plug-ins installation wizard configures the Web server to use the plugin-cfg.xml file that will be created within the application server profile when you run the script. The deployment manager regenerates the plugin-cfg.xml file in the *profile_root*/config/cells/*cell_name*/nodes/*node_name*/servers/*web_server_name* directory. Regeneration occurs whenever a change occurs in the application server configuration that affects deployed applications on the managed node.

After installing the binary plug-in for the local Web server, you must run the script before you can start the Web server. The Web server has already been configured to use the plugin-cfg.xml file in the application server configuration. That file does not exist until you run the configure*web_server_name* script.

Four configurations qualify for the local distributed application server scenario:

Profile type	Federation status	Creation of Web server definition?	Web server already defined in Application Server configuration?
Default application server profile	Federated	By script	N/A
Default Custom profile	Not federated	By script	N/A
Default Custom profile	Federated	By script	N/A
Default deployment manager profile with a managed node (non-default distributed profile)	N/A	By script	N/A

The following overview shows the procedure for completing the configuration and verifying the Web server configuration:

1. Start the deployment manager.
2. If you are planning to add an application server node into a deployment manager cell but have not done so yet, federate the node before installing the plug-ins. If the Web server definition exists when you federate the node, the Web server definition is lost when you federate.
3. Create the Web server definition in the application server. You have two options:
   - Use the administrative console of the deployment manger to create a Web server definition for a managed node. Click **Servers > Web servers > New** and use the Create new Web server entry wizard to create the Web server definition.

- Run the script to manually create the Web server definition within the configuration of the deployment manager. Run the script from the *plugins_root*/bin directory. The script can address the deployment manager on the same machine.

  Open a command window to run the appropriate script:
  – ./configure*web_server_name*

  If you have enabled security or changed the default JMX connector type, edit the script and include the appropriate parameters.

4. Start the Web server with the proper procedure for your Web server.

   For example, start the IBM HTTP Server from a command line:
   - `./IHS_root/bin/apachectl start`

5. Start the application server.

   Change directories to the *profile_root*/bin directory and run the startServer command:
   - `./profile_root/bin/startServer server1`

6. Open the administrative console of the deployment manager. Wait for node synchronization to occur and save the changed configuration that includes the new Web server definition.
7. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the application server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the Web server plug-in.
8. Verify that both Web addresses display the Snoop Servlet - Request/Client Information page.

Scenario 3. Local stand-alone plug-in configuration

The Plug-ins installation wizard creates a Web server definition within the application server profile.

The Plug-ins installation wizard configures the Web server to use the plugin-cfg.xml file that is within the application server profile. The stand-alone application server regenerates the *profile_root*/config/cells/*cell_name*/nodes/*web_server_name*_node/servers/*web_server_name*/plugin-cfg.xml file whenever a change occurs in the application server configuration that affects deployed applications.

After installing the binary plug-in for the local Web server, you can start the Application Server and the Web server immediately upon completion of the installation.

Suppose that you create a Web server definition on a stand-alone application server and then federate the node. The Web server definition is not federated into the cell because the Web server definition is defined as a separate node in a stand-alone Application Server. You must recreate the Web server definition on the managed node. See Scenario 2.

Only one configuration qualifies for the local stand-alone application server scenario:

Profile type	Federation status	Automatic creation of Web server definition?	Web server already defined in Application Server configuration?
Application server	Not federated	Yes	No

Redirection to Scenario 1

An unfederated default standalone application server that has an existing Web server definition is processed as a remote plug-in configuration.

An existing Web server definition on a stand-alone application server causes the Plug-ins installation wizard to follow the remote installation path. A stand-alone application server can have just one Web server definition. Specify the same nick name for the Web server if you want to configure a new Web server.

You can use the plugin-cfg.xml file that is within the Web server definition in the configuration of the Application Server. Simply click **Browse** on the appropriate panel in the Plug-ins installation wizard to select the file. This file must exist. Otherwise, the Plug-ins installation wizard displays a warning and prevents you from proceeding until you select an existing file. The Web server is configured to use this existing plugin-cfg.xml file.

See Scenario 1 for a description of this type of node.

Redirection to Scenario 2

A federated default standalone Application Server is processed as a local distributed plug-in configuration. See Scenario 2 for a description of this type of node.

Overview of the verification procedure

The following overview shows the procedure for verifying the Web server configuration after installing the binary plug-in module:

1. Start the Web server with the proper procedure for your Web server.

   For example, start the IBM HTTP Server from a command line:
   - ./*IHS_root*/bin/apachectl start
2. Start the application server.

   Change directories to the *profile_root*/bin directory and run the startServer command:
   - `./profile_root/bin/startServer server1`

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

Summary

Three scenarios exist for Web server plug-ins for WebSphere Application Server. Each scenario revolves around a unique location for the plug-in configuration file, plugin-cfg.xml. The application server generates the plug-in configuration file. The purpose of the file is to publish the location of all of the application server elements that are relevant to a Web server. Such elements include applications, virtual hosts for serving applications, clusters, and cluster members, for example.

If the Web server cannot get to the file on the application server machine, you must take the file to the Web server. That process is called propagation. Propagation is reserved for the remote plug-in configuration scenario, which is **Scenario 1** in this topic.

In each of the local scenarios, the Web server can get to the plugin-cfg.xml file because it is on the same machine as the file. Two local scenarios exist because of two distinct locations for a local plugin-cfg.xml file.

The configuration scheme for Version 6 of WebSphere Application Server puts the plug-in configuration file in a Web server definition that is either within a Web server node or a managed node. The type of node is the difference between Scenario 2 and Scenario 3 in this topic. All **Scenario 2** configurations require the Web server definition to exist within a managed application server node. All **Scenario 3** configurations have the Web server definition within its own Web server node.

Limited management options do not let you create or delete the one Web server definition in the administrative console of a stand-alone application server. The inability of a stand-alone application server

to create a Web server definition is the basis for the configuration scripts created by the Web server plug-ins for WebSphere Application Server. Without the scripts you could not easily create a Web server definition on a stand-alone application server node.

The location of the plugin-cfg.xml file for each configuration described in this topic is shown in the following table:

Table 11. Plug-in configuration file locations

Scenario	Profile type	Location of the plugin-cfg.xml file		
		Plugins_ install_ root	**profiles_ root: within the managed node**	**profiles_ root: within the Web server node**
1	Any profile anywhere if you select a remote installation type in the Plug-ins installation wizard	X		
	No default profile detected	X		
	Default unfederated (stand-alone) Application Server profile with an existing Web server definition	X		
	Default deployment manager profile with no managed nodes	X		
2	Default application server profile		X	
	Default custom profile		X	
	Default deployment manager profile with a managed node (non-default distributed profile)		X	
3	Default application server profile			X

Legend:

plugins_root

plugins_root

```
/config/
web_server_name/plugin-cfg.xml
```

profiles_ root: within the managed node

```
profile_root
/config/cells/cell_name/nodes/
node_name_of_AppServer/servers/
web_server_name/plugin-cfg.xml
```

profiles_ root: within the Web server node

```
profile_root
/config/cells/cell_name/nodes/
web_server_name_node/servers/
web_server_name/plugin-cfg.xml
```

Web server configuration

Plug-in configuration involves configuring the Web server to use the binary plug-in module that WebSphere Application Server provides. Plug-in configuration also includes updating the plug-in XML configuration file to reflect the current application server configuration. The binary module uses the XML file to help route Web client requests.

The plug-ins configuration process uses the following files to configure a plug-in for the Web server that you select:

- The **Web server configuration file** on the Web server machine, such as the httpd.conf file for IBM HTTP Server.

- The **binary Web server plug-in file** that the installation process installs on the Web server machine.
- The **plug-in configuration file, plugin-cfg.xml**, on the application server machine that you propagate (copy) to a Web server machine.
- The **configuration script** for configuring the Web server definition for your application server in a remote HTTP scenario.

See the following descriptions of each file.

Web server configuration file

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

Configuration consists of adding directives that identify file locations of two files:

- The binary plug-in file
- The plugin-cfg.xml configuration file

The binary Web server plug-in file

See “Web server plug-ins” on page 95 for a description of the binary plug-in module.

An example of a binary plug-in module is the mod_ibm_app_server_http.dll file for IBM HTTP Server on the Windows platform.

Another example of a binary plug-in module is the QSVTAP20 service program on the iSeries platform.

The binary plug-in file does not change. However, the configuration file for the binary plug-in is an XML file. The application server changes the configuration file when certain changes to your WebSphere Application Server configuration occur. See “Web server plug-in configuration service property settings” on page 88 for examples of when the file gets regenerated and when it does not.

The binary module reads the XML file to adjust settings and to route requests to the application server.

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

The plug-in configuration file is an XML file with settings that you can tune in the administrative console. The file lists all of the applications installed on the Web server definition. The binary module reads the XML file to adjust settings and to route requests to the application server.

When you make application server configuration changes that affect deployed applications, regenerate the plug-in configuration XML file.

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

On i5/OS and OS/400 systems, the plug-in is not automatically generated. You must regenerate and propagate the file manually.

See “Web server plug-in configuration service property settings” on page 88 for more information.

The configuration script for the Web server definition

Configuring your Web server with the configureOs400WebserverDefinition script or using the iSeries Administrative GUI creates the configure*web_server_name* script on the Web server machine in the *plugins_root*/bin directory. The script is created for remote installation scenarios only.

Copy the script from the Web server machine to the *app_server_root*/bin directory in the i5/OS partition. Run the script to create a Web server definition in the configuration of the application server.

The iSeries Administrative GUI has plug-ins that allow the administrative console to manage IBM HTTP Servers. Use the administrative console to update your Web server definition with remote Web server management options. Click **Servers > Web servers > *Web_server*** to see configuration options. For example, click **Remote Web server management** to change such properties as:

- Host name
- Administrative port
- User ID
- Password

If a Web server definition already exists for a stand-alone application server, running the script does not add a new Web server definition. Each stand-alone application server can have only one Web server definition.

You cannot use the administrative console of a stand-alone application server to add or delete a Web server definition. However, you can do both tasks using the administrative scripting interface:

- Add a Web server definition through the wsadmin facility using the configure*web_server_name* script. The script uses a Java Command Language (Jacl) script named configureWebserverDefintion.jacl to create and configure the Web server definition.
- Delete a Web server definition using wsadmin commands. The Web server is named webserver1 in the following example:

```
set webserverName webserver1
set webserverNodeSuffix _node
set webserverNodeName
$webserverName$webserverNodeSuffix
$AdminConfig remove
  [$AdminConfig getid
    /Node:$webserverNodeName/Server:$webserverName]
$AdminConfig remove
  [$AdminConfig getid /Node:$webserverNodeName]
$AdminConfig save
```

Alternatively, you can use the configureOs400WebServerDefinition and removeOs400WebServerDefinition scripts to perform these tasks.

A managed node, on the other hand, can have multiple Web server definitions. The script creates a new Web server definition unless the Web server name is the same.

Replacing the default plug-in configuration file with the file from the Web server definition (propagation)

The default file uses fixed parameter values that might not match the parameter values in the actual file on the application server. The default file is a placeholder only.

The file cannot reflect changes that occur in the application server configuration. The file also cannot reflect non-default values that might be in effect on the application server.

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

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

When multiple profiles exist, you can either let the plug-ins installer configure the default profile, or you can select the profile that the plug-ins installer configures. See “Plug-ins configuration” on page 23 for a description of the flow of logic that determines how the installer selects the profile to configure.

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

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

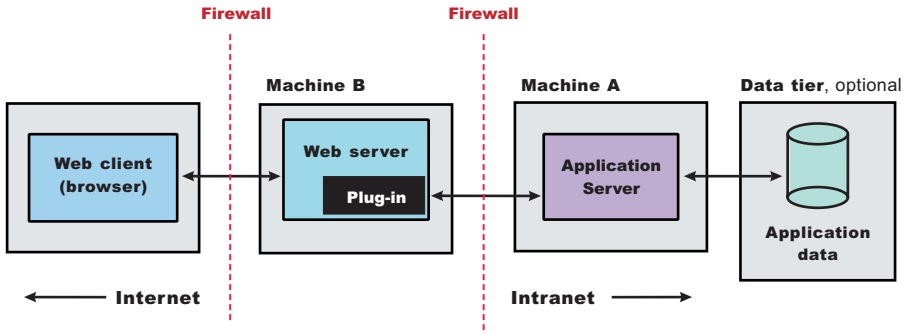
Suppose that you create a new profile. Suppose also that you want to use a Web server. You must install a new Web server for the new profile and use the Plug-ins installation wizard to install the binary plug-in module and to configure both the Web server and the application server.

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

The Plug-ins installation wizard installs the plug-in module, configures the Web server for communicating with the application server, and creates a Web server configuration definition in the application server, if possible.

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

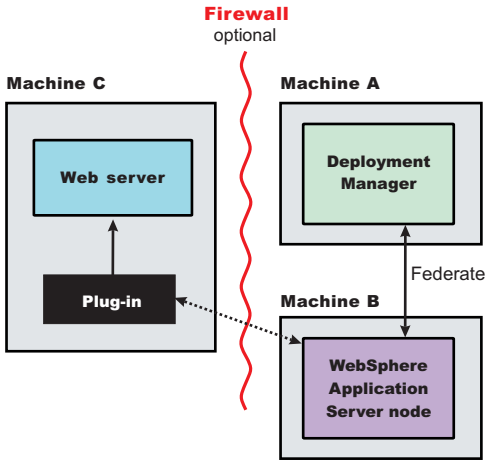
This topic describes how to create the following topology:



Attention:

If you are planning to add the application server node into a deployment manager cell but have not done so yet, start the deployment manager and federate the node before installing the plug-in. You cannot add an application server with a Web server definition into the deployment manager cell.

The following topology is considered a remote topology because the Web server is on a separate machine. The diagram shows a typical remote topology for a distributed environment:



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

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

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

1. Log on to the operating system.

   For example, on some Windows systems, click **Administrative Tools > Local Security Policy > User Rights Assignments** to set the advanced options. See your Windows documentation for more information.
2. Install WebSphere Application Server Network Deployment on Machine A.

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

   Select the Plug-ins installation wizard from the launchpad or change directories to the plugin directory on the product disk or in the downloaded installation image and issue the install command.
5. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.

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

   Press **Ctrl-P** to print the roadmap if the Web browser navigation controls and the menu bar are not present on the browser window that displays the Plug-ins roadmap. Press **Ctrl-W** to close the browser window if the navigation controls and the menu bar do not display. Or close the browser window with the window control in the title bar.
6. Read the license agreement and accept the agreement it if you agree to its terms. Click **Next** when you are finished.

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

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

   - If you stop the installation, see the log file in the /tmp/InstallShield/niflogs directory of the user who installed the plug-ins.
   - If you continue the installation in spite of warnings about missing prerequisites, see the *plugins_root*/logs/install/log.txt file after the installation is complete.

   See Troubleshooting installation for more information about log files.

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

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

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

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

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

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

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

    The default location is shown in “Directory conventions,” on page 471.

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

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

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

    The following list shows configuration files for supported Web servers:

    Apache HTTP Server
    apache_root/config/httpd.conf

    Domino Web Server
    names.nsf and Notes.jar

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

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

    IBM HTTP Server
    IHS_profile_root/conf/httpd.conf

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    For example, on an i5/OS system with an IBM HTTP Server named webserver1 in the default location, copy *plugins_root*/bin/configurewebserver1 from Machine B to the *app_server_root*/bin directory on Machine A.

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

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

    Determine the file encoding and use one of the following procedures to circumvent the failure. To determine the default file encoding, run the appropriate command.Use the result of the command on each machine as the value of the *web_server_machine_encoding* variable and the *application_server_machine_encoding* variable in one of the following procedures.

Procedures for compensating for encoding differences

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

Web server running on a Linux or UNIX system

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

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

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

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

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

Web server running on a Windows machine

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

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

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

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

21. Start the application server on Machine A.

    Use the startServer command, for example:

    - *profile_root*/bin/startServer server1

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

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

    For example: *configurewebserver1.sh Dmgr01 myUserID myPassword*

    The webserver will be configured via wsadmin.

23. From the administrative console of the deployment manager, click **System administration > Save Changes to Master Repository > Synchronize changes with Nodes > Save**.

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

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

    a. Start the Application Server. In a Network Deployment environment, the Snoop servlet is available in the cell only if you included the DefaultApplication when adding the Application Server to the cell. The -includeapps option for the addNode command migrates the DefaultApplication to the cell. If the application is not present, skip this step.

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

       - startServer server1

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

Use either the 2001 page or use the STRTCPSVR SERVER(*HTTP) HTTPSVR(*instance_name*) command to start the IBM HTTP Server.

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

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

d. Verify that snoop is running.

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

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

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

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

Perform the following procedure to correct the error:

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

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

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

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

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

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

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

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

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

See “Selecting a Web server topology diagram and roadmap” on page 12 for an overview of the installation procedure.

See “Web server configuration” on page 30 for more information about the files involved in configuring a Web server.

See “Plug-ins configuration” on page 23 for information about the location of the plug-in configuration file.

See “Editing Web server configuration files” on page 72 for information about how the Plug-ins installation wizard configures supported Web servers.

Configuring multiple Web servers and remote stand-alone application servers

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

When multiple profiles exist, you can either let the plug-ins installer configure the default profile, or you can select the profile that the plug-ins installer configures. See “Plug-ins configuration” on page 23 for a description of the flow of logic that determines how the installer selects the profile to configure.

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

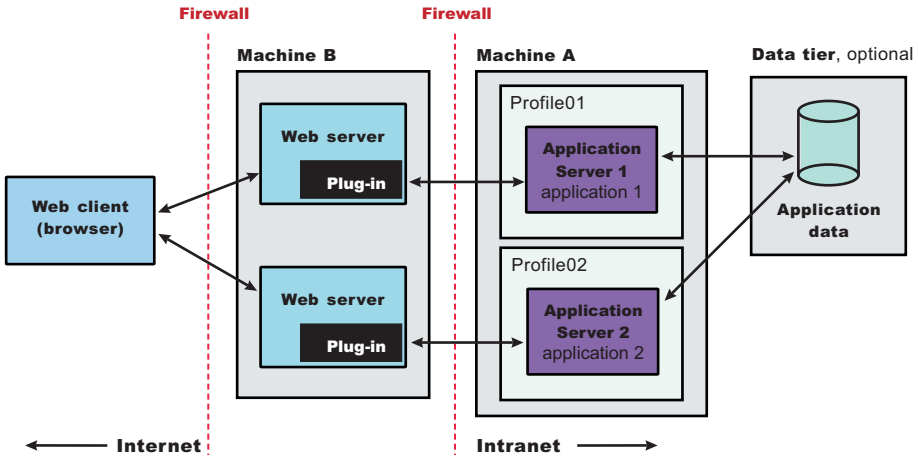
If the WebSphere Application Server product family does not provide a binary plug-in for a particular brand of Web server, then the Web server is not supported. The purpose of the binary plug-in is to provide the communication protocol between the Web server and the application server.

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

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

The Plug-ins installation wizard installs the plug-in module, configures the Web server for communicating with the application server, and creates a Web server configuration definition in the application server, if possible.

This topic describes how to create the following topology:



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

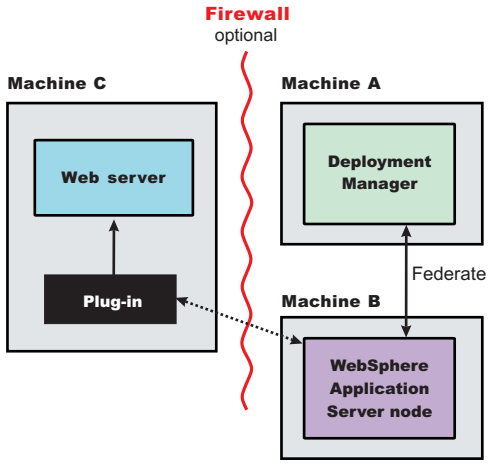
This topology lets each profile have unique applications, configuration settings, data, and log files, while sharing the same set of system files. Creating multiple profiles creates multiple application server environments that you can then dedicate to different purposes.

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

Attention:

If you are planning to add the application server node into a deployment manager cell but have not done so yet, start the deployment manager and federate the node before installing the plug-in. You cannot add an application server with a Web server definition into the deployment manager cell.

The following topology is considered a remote topology because the Web server is on a separate machine. The diagram shows a typical remote topology for a distributed environment:



A deployment manager by itself is also considered a remote scenario if the deployment manager has no managed nodes. Although multiple application servers are not shown in the preceding diagram, Machine B could have more than one application server profile.

1. Log on to the operating system.

   For example, on some Windows systems, click **Administrative Tools > Local Security Policy > User Rights Assignments** to set the advanced options. See your Windows documentation for more information.
2. Install WebSphere Application Server Network Deployment on Machine A.

   See Task overview: installing.
3. Use the managedProfile command to create the first application server profile on Machine A.
4. Install the IBM HTTP Server or another supported Web server on Machine B.
5. Launch the Plug-ins installation wizard on the machine with the Web server.

   Select the Plug-ins installation wizard from the launchpad or change directories to the plugin directory on the product disk or in the downloaded installation image and issue the install command.
6. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.

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

   Press **Ctrl-P** to print the roadmap if the Web browser navigation controls and the menu bar are not present on the browser window that displays the Plug-ins roadmap. Press **Ctrl-W** to close the browser window if the navigation controls and the menu bar do not display. Or close the browser window with the window control in the title bar.
7. Read the license agreement and accept the agreement it if you agree to its terms. Click **Next** when you are finished.
8. If your system does not pass the prerequisites check, stop the installation, correct any problems, and restart the installation. If your system passes the prerequisites check, click **Next**.

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

   - If you stop the installation, see the log file in the /tmp/InstallShield/niflogs directory of the user who installed the plug-ins.

- If you continue the installation in spite of warnings about missing prerequisites, see the *plugins_root*/logs/install/log.txt file after the installation is complete.

See Troubleshooting installation for more information about log files.

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

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

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

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

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

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

    The default location is shown in "Directory conventions," on page 471.

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

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

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

    The following list shows configuration files for supported Web servers:

    Apache HTTP Server
    apache_root/config/httpd.conf

    Domino Web Server
    names.nsf and Notes.jar

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

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

    IBM HTTP Server
    IHS_profile_root/conf/httpd.conf

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

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

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

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

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

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

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

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

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

15. Identify the host name or IP address of Machine A, which is the application server machine, then click **Next**.
16. Examine the summary panel. Click **Next** when you are finished.

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

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

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

    The Web server reads the plugin-cfg.xml file to determine the applications that the application server on Machine A can serve to the Web server on Machine B. Whenever the configuration changes, the application server regenerates the file. When regeneration occurs, propagate, or copy the actual plugin-cfg.xml file from the application server machine to the Web server machine. You can automatically propagate the file to the IBM HTTP Server product.
17. Click **Next** on the pre-installation summary panel to begin the installation or click **Back** to change any characteristics of the installation.

    The panel specifies the plug-ins installation root directory, the Web server plug-ins feature, and the disk size of the code that installs when you click **Next**.
18. After the wizard installs the code and creates the uninstaller program, examine the post-installation summary panel. Click **Next** when you are finished to display the Plug-ins installation roadmap.

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

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

    If a problem occurs and the installation is unsuccessful, examine the logs in the *plugins_root*/logs directory. Correct any problems and reinstall.
19. Close the road map and click **Finish** to exit the wizard.

    Log files from the installation are in the *plugins_root*/logs/install directory.
20. Copy the configure*web_server_name*.sh script on Linux and UNIX systems, the configure*web_server_name*.bat script on Windows systems, or the configure*web_server_name* script on i5/OS from Machine B to the *app_server_root* /bin directory on Machine A.

    For example, on an i5/OS system with an IBM HTTP Server named webserver1 in the default location, copy *plugins_root*/bin/configurewebserver1 from Machine B to the *app_server_root*/bin directory on Machine A.
21. Compensate for file encoding differences to prevent script failure.

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

    Determine the file encoding and use one of the following procedures to circumvent the failure. To determine the default file encoding, run the appropriate command.Use the result of the command on each machine as the value of the *web_server_machine_encoding* variable and the *application_server_machine_encoding* variable in one of the following procedures.

    Procedures for compensating for encoding differences

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

    Web server running on a Linux or UNIX system

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

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

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

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

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

Web server running on a Windows machine

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

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

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

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

22. Start the application server on Machine A.

    Use the startServer command, for example:

    - *profile_root*/bin/startServer server1

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

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

    For example: *configurewebserver1.sh Dmgr01 myUserID myPassword*

    The webserver will be configured via wsadmin.

24. From the administrative console of the deployment manager, click **System administration > Save Changes to Master Repository > Synchronize changes with Nodes > Save**.

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

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

    a. Start the Application Server. In a Network Deployment environment, the Snoop servlet is available in the cell only if you included the DefaultApplication when adding the Application Server to the cell. The -includeapps option for the addNode command migrates the DefaultApplication to the cell. If the application is not present, skip this step.

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

       - startServer server1

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

       Use either the 2001 page or use the STRTCPSVR SERVER(*HTTP) HTTPSVR(*instance_name*) command to start the IBM HTTP Server.

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

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

d. Verify that snoop is running.

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

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

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

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

   Perform the following procedure to correct the error:

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

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

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

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

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

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

28. Use the managedProfile command to create the second application server profile. During the creation process, designate this profile the default profile.

    The script that the Plug-ins installation wizard creates only works on the default profile. So, this script can create only a Web server definition on the profile that is the default profile at the time that the script runs.

29. Install a second IBM HTTP Server or another supported Web server on Machine B.
30. On Machine B, install the Web server plug-ins to configure the second Web server using the Plug-ins installation wizard. Both Web servers share a single installation of the plug-in binaries but must be configured individually.
31. The Plug-ins installation wizard creates a script named configure*web_server_name* for the second Web server. The script is in the *plugins_root*/bin directory on Machine B. Copy the script to the *app_server_root*/bin directory on Machine A.
32. Start the second application server.
33. Run the configure*web_server_name* script to create a Web server definition in the administrative console. You can then use the administrative console to manage the Web server.
34. Propagate the plugin-cfg.xml file from the second application server to the Web server using the administrative console. Click **Servers > Web server > Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.
35. Run the snoop servlet on the second Web server to verify that it is operational.

This procedure results in installing two or more application servers on one machine and installing dedicated Web servers on another machine. This procedure installs the Web server plug-ins for both Web servers and configures both Web servers and both application servers.

See “Selecting a Web server topology diagram and roadmap” on page 12 for an overview of the installation procedure.

See “Editing Web server configuration files” on page 72 for information about how the Plug-ins installation wizard configures supported Web servers.

See “Web server configuration” on page 30 for more information about the files involved in configuring a Web server.

For IHS Web servers, you can stop and start the Web server and propagate the plugin-cfg.xml file from the WebSphere Application Server machine to the Web server machine. For all other Web servers, you can not start/stop or propagate the plugin-cfg.xml file in the admin console. You will need to propagate the plugin-cfg.xml file manually. The following three steps describes how to perform manual propagation:

1. After completion of configuration with Web servers other than IHS 6.x, verify that the plugin-cfg.xml file exists at <WAS_HOME>/profiles/<PROFILE_HOME>/config/cells/<CELL_NAME>/nodes/<SERVER_NAME>/servers/<WEBSERVER_DEFINITION>
2. Transfer the above plugin-cfg.xml to replace <PLUGIN_HOME>/config/<WEBSERVER_DEFINITION>/plugin-xfg.xml
3. Restart the Web server and corresponding profile.

Configuring a Web server and an application server profile on the same machine

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

When multiple profiles exist, you can either let the plug-ins installer configure the default profile, or you can select the profile that the plug-ins installer configures. See “Plug-ins configuration” on page 23 for a description of the flow of logic that determines how the installer selects the profile to configure.

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

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

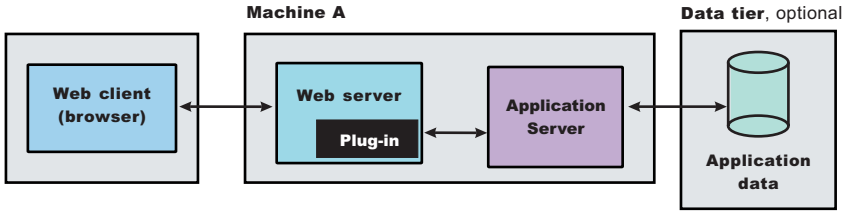
Suppose that you create a new profile. Suppose also that you want to use a Web server. You must install a new Web server for the new profile and use the Plug-ins installation wizard to install the binary plug-in module and to configure both the Web server and the application server.

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

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

However, a stand-alone application server profile and a managed profile can each have multiple Web servers defined, each in a separate Web server definition.

This topic describes how to create the following topology:



The set of steps leading up to the next diagram show how to configure a stand-alone application server. The set of steps after the next diagram show how to configure an application server that is federated into a deployment manager cell.

The wizard performs three steps to properly configure a Web server for Version 6. The wizard performs the steps in the following order:

1. The wizard installs the unique binary plug-in module for the supported Web server after collecting the following information:
   - The type of Web server
   - The location of the configuration file for the Web server that the wizard configures
   - The plug-ins installation root directory for the Web server plug-in modules that the wizard installs
   - The installation root directory of the WebSphere Application Server product, where the wizard creates a Web server definition

   If administrative security is enabled, the Plug-ins installation wizard prompts for the administrative user ID and password for the profile.
2. The wizard prompts you for the location of the configuration file or files for the Web server. You must browse for and select the correct file.

   The wizard edits the configuration file or files for a Web server by creating directives that point to the location of the binary plug-in module and the plug-in configuration file.

   The name of the binary plug-in module varies per Web server type. The plug-in configuration file is always the plugin-cfg.xml file.

3. The wizard creates a Web server definition in the configuration of the application server unless one already exists.

   You can use the administrative console to manage the Web server configuration. For example, when you install an application on the application server, you can also choose to install it on the Web server definition. If so, the updated plugin-cfg.xml file shows that the new application is available. When the Web server reads the updated plug-in configuration file, the Web server becomes aware of the new application that it can serve to Web clients.

   If you choose not to install the new application on the Web server definition, the application is not added to the plug-in configuration file. The Web server is not aware of the application and cannot serve it to Web clients.

Use the following procedure to install the Web server plug-in, configure the Web server, and create a Web server definition in the default application server profile.

1. Log on to the operating system.

   For example, on some Windows systems, click **Administrative Tools > Local Security Policy > User Rights Assignments** to set the advanced options. See your Windows documentation for more information.
2. Install WebSphere Application Server Network Deployment on Machine A.

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

   Select the Plug-ins installation wizard from the launchpad or change directories to the plugin directory on the product disk or in the downloaded installation image and issue the install command.
5. Stop the stand-alone application server before installing the Web server plug-ins. For example, assuming that the profile name is default, use one of the following commands.
   - /usr/ IBM/ WebSphere/ AppServer/ profiles/ *default*/ bin/ stopServer server1
6. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.

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

   Press **Ctrl-P** to print the roadmap if the Web browser navigation controls and the menu bar are not present on the browser window that displays the Plug-ins roadmap. Press **Ctrl-W** to close the browser window if the navigation controls and the menu bar do not display. Or close the browser window with the window control in the title bar.
7. Read the license agreement and accept the agreement it if you agree to its terms. Click **Next** when you are finished.
8. If your system does not pass the prerequisites check, stop the installation, correct any problems, and restart the installation. If your system passes the prerequisites check, click **Next**.

   Look for the appropriate log file for information about missing prerequisites:
   - If you stop the installation, see the log file in the /tmp/InstallShield/niflogs directory of the user who installed the plug-ins.
   - If you continue the installation in spite of warnings about missing prerequisites, see the *plugins_root*/logs/install/log.txt file after the installation is complete.

   See Troubleshooting installation for more information about log files.
9. Select the type of Web server that you are configuring and click **Next**.

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

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

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

10. Select **Application Server machine (local)** and click **Next**.
11. Accept the default location for the installation root directory for the plug-ins. Click **Next**.

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

    The default location is shown in “Directory conventions,” on page 471.

    A possibility exists that the Web server might run on a platform that WebSphere Application Server does not support.
12. Click **Browse** on the Application Server Installation Location panel to browse for the location of the application server profile if necessary. Click **Next** when the installation root directory is correct.

    The fully qualified path identifies the installation root directory for the WebSphere Application Server product, which is referred to as the app_server_root throughout the information center.
13. Enter an administrative user ID and password if administrative security is enabled on the application server. If more than one profile is created under the defined Application Server installation, a panel is displayed that you can use to select a profile to configure. The selected profile becomes the default profile. If only one profile exists, the default profile is automatically selected and this panel does not appear.
14. Click **Browse** to select the configuration file for your Web server, verify that the Web server port is correct, and then click **Next** when you are finished.

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

    The following list shows configuration files for supported Web servers:

    Apache HTTP Server
    apache_root/config/httpd.conf

    Domino Web Server
    names.nsf and Notes.jar

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

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

    IBM HTTP Server
    IHS_profile_root/conf/httpd.conf

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

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

    The wizard displays a naming panel for the nickname of the Web server definition.
15. Specify a nickname for the Web server. Click **Next** when you are finished.

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

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

    ```
    $AdminTask deleteServer { -serverName webserver1 -nodeName webserver1_node }
    $AdminTask removeUnmanagedNode { -nodeName webserver1_node }
    $AdminConfig save
    ```
16. Specify the location for the plugin-cfg.xml file and click **Next**.

    This is a critical selection.

See “Plug-ins configuration” on page 23 for a description of the logic that determines what path is configured by default. The following possibilities exist for the default location of the plug-in configuration file. The wizard determines the characteristics of the application server to determine the best path for the file:

- An application server that has an existing Web server definition has the following path:

  plugins_profile_root/config/
  web_server_name/plugin-cfg.xml

- A stand-alone application server that does not have a Web server definition has the following path:

  profile_root
  /config/cells/*cell_name*/nodes/
  *web_server_name*_node/servers/
  web_server_name/plugin-cfg.xml

You can accept the default value if the application server does not have a Web server definition.

Using an existing Web server definition

If the application server has a Web server definition, the wizard cannot create a new Web server definition within the application server configuration. However, the wizard can reconfigure the Web server. Click **Browse** and select the existing plugin-cfg.xml file in the application server configuration.

To find the plug-in configuration file in a stand-alone application server, follow this file path:

profile_root
/config/cells/*cell_name*/nodes/
*web_server_name*_node/servers/
web_server_name/plugin-cfg.xml

If the existing *web_server_name* is different than the nickname that you gave the Web server in the wizard, click **Back** to return to the naming panel for the Web server and change the name to match the existing Web server definition name.

If you cannot find an existing plugin-cfg.xml file after all, you must install the temporary plugin-cfg.xml file. In such a case, type the path to the plug-ins installation root directory so that the wizard can install the temporary plug-in configuration file:

plugins_root/config/
web_server_name/plugin-cfg.xml

17. Click **Next** after verifying the characteristics of the plug-ins installation or click **Back** to make changes.

    Once created, a Web server definition on a stand-alone application server node cannot be removed except through scripting. (See “Uninstalling the Web server plug-ins for WebSphere Application Server” on page 77 for the procedure.)

    You can, however, reuse the same definition for a different type of Web server. Run the Plug-ins installation wizard to configure a new Web server in that situation. The Plug-ins installation wizard configures the new Web server to use the existing plugin-cfg.xml file.

18. Click **Next** on the pre-installation summary panel to begin the installation or click **Back** to change any characteristics of the installation. The wizard begins installing the plug-ins and configuring the Web server and the application server.

    The wizard shows an installation status panel as it installs the plug-ins.

    The wizard displays the Installation summary panel at the completion of the installation.

19. Verify the success of the installation on the Installation summary panel and click **Finish** to exit the wizard.

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

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

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

a. Start the Application Server. In a Network Deployment environment, the Snoop servlet is available in the cell only if you included the DefaultApplication when adding the Application Server to the cell. The -includeapps option for the addNode command migrates the DefaultApplication to the cell. If the application is not present, skip this step.

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

   - startServer server1

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

   Use either the 2001 page or use the STRTCPSVR SERVER(*HTTP) HTTPSVR(*instance_name*) command to start the IBM HTTP Server.

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

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

d. Verify that snoop is running.

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

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

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

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

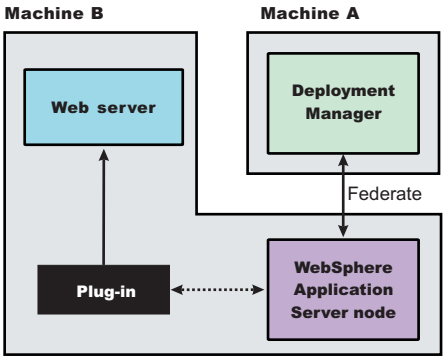
   Perform the following procedure to correct the error:

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

21. Configure a Web server and a distributed application server profile on the same machine.

    The rest of these steps describe how to configure an application server that is federated into a deployment manager cell.

    The following topology is considered a local distributed topology because it involves a cell:



This part of the procedure assumes that you have already installed the Network Deployment product on both machines. Also assumed is that you have already configured a deployment manager profile on Machine A and an application server profile on Machine B.

If you are planning to add the application server node into a deployment manager cell but have not done so yet, start the deployment manager and federate the node before installing the plug-in. You cannot add an application server with a Web server definition into the deployment manager cell.

A Web server definition on a federated application server is installed on the same managed node as the application server. There is one node, but with two server processes, the application server and the Web server definition.

If you are installing the plug-ins for use with a federated application server, start the deployment manager. Verify that the node agent process on the managed node is also running. Both the deployment manager and the node agent must be running to successfully configure a managed node.

22. Install IBM HTTP Server or another supported Web server on Machine B.
23. Launch the Plug-ins installation wizard on the machine with the Web server.
24. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.
25. Read the license agreement and accept the agreement it if you agree to its terms, then click **Next**.
26. If your system does not pass the prerequisites check, stop the installation, correct any problems, and restart the installation. If your system passes the prerequisites check, click **Next**.
27. Select the type of Web server that you are configuring, then click **Next**.
28. Select **Application Server machine (local)** and click **Next**.
29. Accept the default location for the installation root directory for the plug-ins, then click **Next**.
30. Click **Browse** on the Application Server installation location panel to browse for the location of the Application Server profile, if necessary. Click **Next** when the installation root directory is correct.
31. Click **Browse** to select the configuration file for your Web server, verify that the Web server port is correct, and then click **Next**.
32. Specify a nickname for the Web server, then click **Next**.
33. Specify the location for the plugin-cfg.xml file and click **Next**.

    This is a critical selection. A federated application server that does not have a Web server definition has the following path:

    ```
    profile_root
    /config/cells/cell_name/nodes/
    node_name_of_AppServer/servers/
       web_server_name/plugin-cfg.xml
    ```

    An application server that has an existing Web server definition has the following path:

    ```
    plugins_root/config/
       web_server_name/plugin-cfg.xml
    ```

    See “Plug-ins configuration” on page 23 for a description of the logic that determines what path is configured by default.

34. Click **Next** after verifying the characteristics of the plug-ins installation or click **Back** to make changes.

    You can use the administrative console of the deployment manger to delete an existing Web server or to create new ones. Federated nodes can have more than one Web server definition.
35. Click **Next** on the pre-installation summary panel to begin the installation or click **Back** to change any characteristics of the installation.

    The wizard begins installing the plug-ins and configuring the Web server and the application server.

    The wizard shows an installation status panel as it installs the plug-ins.

    The wizard displays the Installation summary panel at the completion of the installation.
36. Verify the success of the installation on the Installation summary panel and click **Finish** to exit the wizard.
37. Complete the installation by creating the Web server definition.

    You can use the administrative console of the deployment manager to create the Web server definition on a federated node. Or, you can run the configuration script that the Plug-ins installation wizard created.

    The script already contains all of the information that you must gather when using the administrative console option.

    Select one of the following options:

    - **Using the administrative console**

      Click **Servers > Web servers > New** and use the Create new Web server entry wizard to create the Web server definition.
    - **Running the configuration script**

      Issue the appropriate command from a command window:
      – `./plugins_root/bin/configureweb_server_name`

      If you have enabled security or changed the default JMX connector type, edit the script and include the appropriate parameters on the wsadmin command.
38. From the administrative console of the deployment manager, click **System administration > Save Changes to Master Repository > Synchronize changes with Nodes > Save**.
39. Source the Domino Web server script if necessary.
40. Start the snoop servlet.

    See the snoop procedure for the stand-alone application server for the full procedure.

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

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

The Plug-ins installation wizard creates a Web server definition within the application server profile unless one already exists.

The Plug-ins installation wizard configures the Web server to use the *profile_root*/plugin-cfg.xml file.

The application server regenerates the Web server plug-in configuration file, plugin-cfg.xml whenever an event occurs that affects the file. Such events include the addition or removal of an application, server, or virtual host. The stand-alone application server regenerates the file in the following location:

profile_root
   /config/cells/*cell_name*/nodes/
   *web_server_name*_node/servers/
   web_server_name/plugin-cfg.xml

On a federated node, the creation or removal of clusters and cluster members also causes file regeneration. The deployment manager regenerates the file for a federated application server in the following location:

profile_root
   /config/cells/*cell_name*/nodes/
   node_name_of_AppServer/servers/
   web_server_name/plugin-cfg.xml

You can start a stand-alone application server and the Web server immediately after installing of the binary plug-in for the local Web server. Open the administrative console of the application server after you start the server and save the changed configuration.

After installing the binary plug-in for the local Web server, you can start a federated application server and the Web server after running the configuration script that completes the installation. Open the administrative console of the deployment manager. Wait for node synchronization to occur. Save the changed configuration that includes the new Web server definition.

See “Selecting a Web server topology diagram and roadmap” on page 12 for an overview of the installation procedure.

See “Plug-ins configuration” on page 23 for information about the location of the plug-in configuration file.

See “Web server configuration” on page 30 for information about the files involved in configuring a Web server.

See “Editing Web server configuration files” on page 72 for information about how the Plug-ins installation wizard configures supported Web servers.

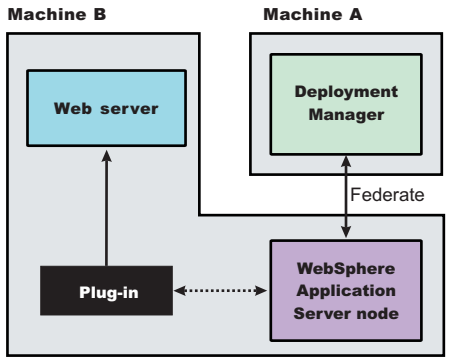
See “Installing Web server plug-ins” on page 9 for information about other installation scenarios for installing Web server plug-ins.

Configuring a Web server and a custom profile on the same machine

This procedure describes installing a Web server and its plug-in on a machine where the default profile is a custom profile.

When multiple profiles exist, you can either let the plug-ins installer configure the default profile, or you can select the profile that the plug-ins installer configures. See “Plug-ins configuration” on page 23 for a description of the flow of logic that determines how the installer selects the profile to configure.

This procedure configures the custom profile that is the default profile on the machine. This procedure assumes that you already have installed a deployment manger on Machine A.



The WebSphere Application Server node on Machine B is the custom node that you create in this procedure. This procedure starts the deployment manager and federates the custom node before installing the Web server plug-ins.

Start the deployment manager. The deployment manager must be running to successfully federate and configure the custom node.

Use the following procedure to install the Web server plug-in, configure the Web server, and create a Web server definition in the default custom profile (custom node).

1. Log on to the operating system.

   For example, on some Windows systems, click **Administrative Tools > Local Security Policy > User Rights Assignments** to set the advanced options. See your Windows documentation for more information.
2. Install the WebSphere Application Server Network Deployment product.

   See Task overview: installing.
3. Create a custom profile as the first profile on the machine and federate the node as you create it.
4. **Optional:** Use the administrative console of the deployment manager to create an Application Server on the custom node.

   Click **Servers > Applications servers > New** and follow the instructions to create a server. A server is not required for installing the plug-ins but it lets you verify the functionality of the Web server.
5. **Optional:** Install the DefaultApplication on the new server while you are in the administrative console of the deployment manager.

   The DefaultApplication includes the snoop servlet. The verification step uses the snoop servlet.
6. Stop the application server if it is running.

   Use the stopServer command or the administrative console of the deployment manager to stop the managed node.
7. Install the IBM HTTP Server or another supported Web server on Machine B.
8. Launch the Plug-ins installation wizard on the machine with the Web server.

   Select the Plug-ins installation wizard from the launchpad or change directories to the plugin directory on the product disk or in the downloaded installation image and issue the install command.
9. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.

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

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

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

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

    - If you stop the installation, see the log file in the /tmp/InstallShield/niflogs directory of the user who installed the plug-ins.
    - If you continue the installation in spite of warnings about missing prerequisites, see the *plugins_root*/logs/install/log.txt file after the installation is complete.

    See Troubleshooting installation for more information about log files.
12. Select the type of Web server that you are configuring and click **Next**.

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

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

    If you select the Web server identification option labeled **None**, the Web server installs the binary plug-ins but does not configure the Web server.
13. Select **Application server machine (local)** and click **Next**.
14. Accept the default location for the installation root directory for the plug-ins. Click **Next**.

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

    The default location is shown in “Directory conventions,” on page 471.

    A possibility exists that the Web server might run on a platform that WebSphere Application Server does not support.
15. Click **Browse** on the Application Server installation location panel to browse for the location of the managed node, if necessary. Click **Next** when the installation root directory is correct.

    The fully qualified path identifies the installation root directory for the Network Deployment product core files.
16. Click **Browse** to select the configuration file for your Web server, verify that the Web server port is correct, and then click **Next** when you are finished.

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

    The following list shows configuration files for supported Web servers:

    Apache HTTP Server
    apache_root/config/httpd.conf

    Domino Web Server
    names.nsf and Notes.jar

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

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

    IBM HTTP Server
    IHS_profile_root/conf/httpd.conf

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

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

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

17. Specify a nick name for the Web server and click **Next**.

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

18. Accept the location for the plugin-cfg.xml file and click **Next**.

    See "Plug-ins configuration" on page 23 for a description of the logic that determines what path is configured by default. The wizard determines the characteristics of the managed application server node to determine the best path for the file:

    A federated custom node has the following path:

    profile_root
    /config/cells/*cell_name*/nodes/
    node_name_of_custom_profile/servers/
    web_server_name/plugin-cfg.xml

    profile_root
    /config/cells/*cell_name*/nodes/
    node_name_of_custom_profile/servers/
    web_server_name/plugin-cfg.xml

    Accept the default value.

19. Click **Next** after verifying the characteristics of the plug-ins installation or click **Back** to make changes.

    You can use the administrative console of the deployment manger to delete an existing Web server or to create new ones. Federated nodes can have more than one Web server definition.

20. Click **Next** on the pre-installation summary panel to begin the installation or click **Back** to change any characteristics of the installation. The wizard begins installing the plug-ins and configuring the Web server and the managed custom node.

    The wizard shows an installation status panel as it installs the plug-ins. The wizard displays the Installation summary panel at the completion of the installation.

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

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

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

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

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

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

23. Complete the installation by creating the Web server definition.

    You can use the administrative console of the deployment manager to create the Web server definition on a federated node. Or, you can run the configuration script that the Plug-ins installation wizard created.

    The script already contains all of the information that you must gather when using the administrative console option.

    Select one of the following options:

    - **Using the administrative console**

      Click **Servers > Web servers > New** and use the Create new Web server entry wizard to create the Web server definition.

    - **Running the configuration script**

      Issue the appropriate command from a command window:
      – *plugins_root*/bin/configure*web_server_name*

If you have enabled security or changed the default JMX connector type, edit the script and include the appropriate parameters on the wsadmin command.

Shell scripts on some operating systems cannot contain double-byte characters or single-byte characters with pronunciation keys.

If the file path to the Web server includes double-byte characters or single-byte characters with pronunciation keys, such as o-umlaut (a diacritic mark over the o), c-cedilla (a mark is under the c), or characters with other keys, the script might not run correctly. Copy the content of such a script to the command line and run the wsadmin command directly.

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

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

    a. Start the Application Server. In a Network Deployment environment, the Snoop servlet is available in the cell only if you included the DefaultApplication when adding the Application Server to the cell. The -includeapps option for the addNode command migrates the DefaultApplication to the cell. If the application is not present, skip this step.

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

       - startServer server1

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

       Use either the 2001 page or use the STRTCPSVR SERVER(*HTTP) HTTPSVR(*instance_name*) command to start the IBM HTTP Server.

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

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

    d. Verify that snoop is running.

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

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

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

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

       Perform the following procedure to correct the error:

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

6) If you still have problems, check the IBM HTTP Server admin_error.log file and the WebSphere Application Server logs (trace.log file) to determine the cause of the problem.

25. If the deployment manager does not have the DefaultApplication installed, you can test the functionality of the Web server and the custom node using an application of your own.
26. From the administrative console of the deployment manager, click **System administration > Save Changes to Master Repository > Synchronize changes with Nodes > Save**.
27. To create multiple Web server definitions for the managed node, use the Plug-ins installation wizard to configure each Web server.

    Identify the same managed node each time. Give each Web server a different nick name.

This procedure results in the installation of the Web server plug-ins for WebSphere Application Server on a Web server machine. The Plug-ins installation wizard creates a Web server definition within the managed node.

The Plug-ins installation wizard configures the Web server to use the plugin-cfg.xml file that is within the managed custom node.

The deployment manager regenerates the Web server plug-in configuration file, plugin-cfg.xml whenever an event occurs that affects the file. Such events include the addition or removal of an application, server, or virtual host.

The creation or removal of clusters and cluster members also causes file regeneration. Automatic propagation through node synchronization copies the file after each regeneration to the following location on the custom node machine:

profile_root
/config/cells/*cell_name*/nodes/
node_name_of_custom_profile/servers/
web_server_name/plugin-cfg.xml

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

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

After installing the binary plug-in for the local Web server, you can start the managed node and the Web server after running the configuration script that completes the installation.

See “Plug-ins configuration” on page 23 for information about the location of the plug-in configuration file.

See “Web server configuration” on page 30 for more information about the files involved in configuring a Web server.

See “Editing Web server configuration files” on page 72 for information about how the Plug-ins installation wizard configures supported Web servers.

See “Installing Web server plug-ins” on page 9 for information about other installation scenarios for installing Web server plug-ins.

Configuring a Web server and a deployment manager profile on the same machine

This procedure describes installing a Web server and its plug-in on a machine where the default profile is a deployment manager.

When multiple profiles exist, you can either let the plug-ins installer configure the default profile, or you can select the profile that the plug-ins installer configures. See “Plug-ins configuration” on page 23 for a description of the flow of logic that determines how the installer selects the profile to configure.

This procedure configures the deployment manager profile that is the default profile on the machine. A managed node must exist to define a Web server definition, which is always on a managed node. If the deployment manager is the default profile, the Plug-ins installation wizard looks for a managed custom node in the deployment manager configuration. If the deployment manager does not have a managed custom node, the Plug-ins installation wizard looks for a managed application server node. If the deployment manager does not have a managed node, then the Plug-ins installation wizard classifies the installation as a remote installation.

Start the deployment manager and the node agent for the managed node. The deployment manager and the node must be running to successfully change its configuration.

Use the following procedure to install the Web server plug-in, configure the Web server, and create a Web server definition in the default profile.

1. Log on to the operating system.

   For example, on some Windows systems, click **Administrative Tools > Local Security Policy > User Rights Assignments** to set the advanced options. See your Windows documentation for more information.
2. Install the WebSphere Application Server Network Deployment product.

   See Task overview: installing.
3. Create a deployment manager profile as the first profile on the machine.
4. Install the IBM HTTP Server or another supported Web server.
5. Launch the Plug-ins installation wizard on the machine with the Web server.

   Select the Plug-ins installation wizard from the launchpad or change directories to the plugin directory on the product disk or in the downloaded installation image and issue the install command.
6. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.

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

   Press **Ctrl-P** to print the roadmap if the Web browser navigation controls and the menu bar are not present on the browser window that displays the Plug-ins roadmap. Press **Ctrl-W** to close the browser window if the navigation controls and the menu bar do not display. Or close the browser window with the window control in the title bar.
7. Read the license agreement and accept the agreement it if you agree to its terms. Click **Next** when you are finished.
8. If your system does not pass the prerequisites check, stop the installation, correct any problems, and restart the installation. If your system passes the prerequisites check, click **Next**.

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

   - If you stop the installation, see the log file in the /tmp/InstallShield/niflogs directory of the user who installed the plug-ins.
   - If you continue the installation in spite of warnings about missing prerequisites, see the *plugins_root*/logs/install/log.txt file after the installation is complete.

   See Troubleshooting installation for more information about log files.
9. Select the type of Web server that you are configuring and click **Next**.

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

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

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

10. Select **Application Server machine (local)** and click **Next**.
11. Accept the default location for the installation root directory for the plug-ins. Click **Next**.

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

    The default location is shown in "Directory conventions," on page 471.

    A possibility exists that the Web server might run on a platform that WebSphere Application Server does not support.
12. Click **Browse** on the Application Server installation location panel to browse for the location of the deployment manager, if necessary. Click **Next** when the installation root directory is correct.

    The fully qualified path identifies the installation root directory for the Network Deployment product core files.
13. Click **Browse** to select the configuration file for your Web server, verify that the Web server port is correct, and then click **Next** when you are finished.

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

    The following list shows configuration files for supported Web servers:

    Apache HTTP Server
    apache_root/config/httpd.conf

    Domino Web Server
    names.nsf and Notes.jar

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

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

    IBM HTTP Server
    IHS_profile_root/conf/httpd.conf

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

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

    The wizard displays a naming panel for the nickname of the Web server definition.
14. Specify a nickname for the Web server and click **Next**.

    The wizard uses the value to name configuration folders in the plug-ins installation root directory. The wizard also uses the name within the deployment manager as the name of the Web server definition.
15. Accept the location for the plugin-cfg.xml file and click **Next**.

    See "Plug-ins configuration" on page 23 for a description of the logic that determines what path is configured by default. The wizard determines the characteristics of the deployment manager to determine the best path for the file.

    When the deployment manager is the default profile, the path is:

    plugins_profile_root/config/
    web_server_name/plugin-cfg.xml

    Accept the default value.

Important: If there is a managed custom node on the deployment manager machine, the Plug-ins installation wizard uses the following file path:

```
profile_root
   /config/cells/cell_name/nodes/
   node_name_of_custom_profile/servers/
   web_server_name/plugin-cfg.xml
```

In this case, accept the path and resume the procedure at this point in “Configuring a Web server and a custom profile on the same machine” on page 54.

16. Click **Next** after verifying the characteristics of the plug-ins installation or click **Back** to make changes.

    You can use the administrative console of the deployment manger to delete an existing Web server or to create new ones. Federated nodes can have more than one Web server definition.

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

    The wizard begins installing the plug-ins and configuring the Web server and the deployment manager.

    The wizard shows an installation status panel as it installs the plug-ins.

    The wizard displays the Installation summary panel at the completion of the installation.

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

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

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

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

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

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

20. Complete the installation by creating the Web server definition.

    You must create an application server profile or a custom profile and federate the node before you can use the administrative console of the deployment manager to create a Web server definition. The same is true for running the configuration script that the Plug-ins installation wizard created. You must assign the Web server to a managed node when you create it.

    The managed node must exist before running the Plug-ins installation wizard. Otherwise, the installation is considered a remote installation.

    If you install the plug-in, save the script to run after you create a managed node. Otherwise an error occurs. Before starting the Web server, wait for these actions to occur:

    - The script runs successfully.
    - The script creates the Web server definition on the managed node.
    - Node synchronization occurs.

    Adding the node starts the nodeagent process. If the node agent is not running for some reason, start the node.

    Tip: If you want the Web server to handle requests for an application for multiple managed nodes, install the application on each managed node and on the Web server definition.

    The script already contains all of the information that you must gather when using the administrative console option.

    Select one of the following options:

    - **Using the administrative console**

Click **Servers > Web servers > New** and use the Create new Web server entry wizard to create the Web server definition.

- **Running the configuration script**

  If the node has only a deployment manager profile, then the plug-ins installer reverts to a remote plug-in configuration. You must manually copy the *plugins_root*/ bin/ configure*web_server_name*.sh script or the *plugins_root*\ bin\ configure*web_server_name*.bat script to the *app_server_root*/bin directory of the deployment manager to run the script.

  Issue the appropriate command to configure the Web server.

  If you have enabled security or changed the default JMX connector type, edit the script and include the appropriate parameters on the wsadmin command.

21. From the administrative console of the deployment manager, click **System administration > Save Changes to Master Repository > Synchronize changes with Nodes > Save**.
22. Start the Snoop servlet to verify the ability of the Web server to retrieve an application from the Application Server.

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

    a. Start the Application Server. In a Network Deployment environment, the Snoop servlet is available in the cell only if you included the DefaultApplication when adding the Application Server to the cell. The -includeapps option for the addNode command migrates the DefaultApplication to the cell. If the application is not present, skip this step.

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

       - startServer server1

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

       Use either the 2001 page or use the STRTCPSVR SERVER(*HTTP) HTTPSVR(*instance_name*) command to start the IBM HTTP Server.

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

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

    d. Verify that snoop is running.

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

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

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

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

       Perform the following procedure to correct the error:

       1) Verify that the IBM HTTP Server administration server is running.
       2) Verify that the Web server host name and the port that is defined in the WebSphere Application Server administrative console matches the IBM HTTP Server administration host name and port.
       3) Verify that the fire wall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere Application Server administrative console.
       4) Verify that the user ID and password that is specified in the WebSphere Application Server administrative console under remote managed, is created in the admin.passwd file, using the htpasswd command.

5) If you are trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere Application Server key database as a signer certificate. This key database is specified by the com.ibm.ssl.trustStore directive in the sas.client.props file in the profile where your administrative console is running. This consideration is primarily for self-signed certificates.
6) If you still have problems, check the IBM HTTP Server admin_error.log file and the WebSphere Application Server logs (trace.log file) to determine the cause of the problem.

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

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

The Plug-ins installation wizard configures the Web server to use the *plugins_root*/plugin-cfg.xml file.

After installing the binary plug-in for the local Web server, you must create a managed node before you can successfully run the configuration script and use the Web server.

See “Plug-ins configuration” on page 23 for an overview of the installation procedure.

See “Web server configuration” on page 30 for more information about the files involved in configuring a Web server.

See “Editing Web server configuration files” on page 72 for information about how the Plug-ins installation wizard configures supported Web servers.

See “Installing Web server plug-ins” on page 9 for information about other installation scenarios for installing Web server plug-ins.

responsefile.txt

This topic describes the response file for performing a silent installation of the Web server plug-ins for WebSphere Application Server.

Install the product silently using an options response file.

The responsefile.txt file has directives that set installation options. Comments in the file describe how to set the string value for each directive.

Use the options file to run the Plug-ins installation wizard in silent mode, which is referred to as installing silently. The wizard reads the options file to determine responses and does not display the graphical user interface. Use the following command to use a copy of the options file named myresponsefile.txt for a silent installation:

```
install -options myresponsefile.txt
```

Location of the response file

The sample options response file is named responsefile.txt. The file is in the plugin directory on the product disc or in the downloaded installation image.

Mode of use

The Plug-ins installation wizard can read an existing options response file and run silently without displaying the graphical user interface.

Installing silently

The options file supplies the values to the Plug-ins installation wizard when installing silently. The wizard reads the options file to determine responses and does not display the graphical user interface. Use the following command to use a copy of the options file named myresponsefile.txt for a silent installation:

```
install -options myresponsefile.txt
```

Creating an operational environment

The installation of the plug-ins is a three-step process:

1. Installing the binary plug-in modules for supported Web servers
2. Configuring the Web servers to use the binary module to communicate with the application server
3. Creating a Web server definition in the application server

   As you install an application, you can install it on the Web server definition in addition to the application server. All applications on the Web server definition are listed in its plug-in configuration file. After propagation, the real Web server can access the applications.

The sample options response file, responsefile.txt, controls installing the binary plug-ins, configuring the Web server, and creates a script for creating the Web server definition on a remote application server machine. The script is customized according to values supplied in the responsefile.txt file. The script is generated to run on the application server machine to create the Web server definition.

If the Web server is on the same machine as a stand-alone application server, the responsefile.txt file can create the Web server definition directly without creating a script.

To edit and use the response file for installing the plug-ins and configuring the Web server and application server, perform the following procedure:

1. Copy the responsefile.txt file from the plugins directory on the product disc to a place that you can easily identify on your machine.
2. Edit the file to customize the values for your installation.
3. Save the file.
4. Start the installation. For example:

   ```
   install -options path/myresponsefile.txt
   ```

5. After the installation, examine the logs for success.

Logging

If no installation logs exist, refer to temporary log file, log.txt in your <userhome>/plglogs directory. You can also cause ISMP to record status about a problem that is preventing the installation from occurring, as described in the following section.

For example, if you start the silent installation without accepting the license in the -OPT silentInstallLicenseAcceptance="false" directive, the installation does not occur. The fact that the license entry was not accepted is recorded in log.txt in the <userhome>/plglogs directory.

If any response file validation results in a failure, the failure is listed in the plglogs file, then the installation fails.

If all validations pass, the installation occurs. Then, the Plug-ins installation wizard records installation events in the following log files. The log files are in the *plugins_root*/logs/install directory:

log.txt Records all of the ISMP events that occur during the installation. The log also describes whether

the installation was local or remote. Messages at the end of the file indicate whether manual configuration steps are required to complete the installation.

Key elements to look for in the installation record are:

Manual steps warning

When the wizard requires you to run a script to create the Web server definition, the wizard refers to the fact that manual steps are required.

If manual steps are required, the name and location of the script that you must run are written in the log file at the end of the installation record.

Web server type

The log has a record of the Web server type, such as IHS for the IBM HTTP Server, for example.

Location of the plug-in configuration file

The log has a record of the plugin-cfg.xml file location currently in the Web server configuration.

installconfig.log
Lists all of the configuration events that occur during the installation.

installGSKit.log
Lists events that occur during the installation of the GSKit code.

The command line for the installation is listed when the installation occurs. The GSKit 7 installation record is written after the GSKIT 7 : entry in the log.

install*Web_server_type***Plugin.log**
Records events that occur during the installation of a Web server plug-in. The name of the file varies to reflect the Web server:

- installAPACHEPlugin.log
- installIHSPlugin.log
- installIISPlugin.log
- installSunOnePlugin.log
- installDomino5Plugin.log
- installDomino6Plugin.log
- installDomino7Plugin.log

Each log lists the following critical information:

- The plug-in binary module that is currently installed
- The current location of the plug-in configuration file that is configured for the Web server

configure*_Web_server_type_***webserver.log**
Lists events that occur during the configuration of a Web server plug-in. The name of the file varies to reflect the Web server:

- configure_APACHE_webserver.log
- configure_IHS_webserver.log
- configure_IIS_webserver.log
- configure_SUNJAVASYSTEM_webserver.log
- configure_DOMINO_webserver.log

The configure*Web_server_type*_webserver.log file reports the actions that the Plug-ins installation wizard performs as it updates the Web server configuration file.

In a remote scenario, this log is not present because you must run the script to create the Web server definition manually.

In a federated scenario, the script is created and this log is not present.

Usage notes

- The file is not a read-only file.
- Edit this file directly with your flat file editor of choice, such as WordPad on a Windows platform.
- The file is updated when you specify the -options parameter when using the Plug-ins installation wizard.
- The file must exist to perform a silent installation. The installation program reads this file to determine installation option values when you install silently. Provide the fully qualified file path.
- Save the file in a location that you can identify when you specify the fully qualified path as part of the installation command.

Example responsefile.txt file

Edit the version of the file that ships with your WebSphere Application Server product. The following example is not guaranteed to be an accurate representation of the file that ships with the product.

```
# ********************************************************************
#
# Response file for Web Server Plug-ins for WebSphere Application
# Server V6.1 Install
#
# ********************************************************************

###############################################################################
#
# A common use of an options file is to run the wizard in silent mode. This lets
# the options file's author to specify wizard settings without having to run the
# wizard in graphical or console mode. To use this options file for silent mode
# execution, use the following command line arguments when running the wizard:
#
#   -options "/<responsefile.path>/responsefile"
#
###############################################################################

###############################################################################
#
# Invoke the install wizard in silent mode for both local and remote install
# whenever the response file is used.
#

-silent

###############################################################################
#
# TROUBLE SHOOTING TIP
#
# If no signs of an install are visible, reference the log file (plglogs/log.txt)
# in the /tmp/InstallShield directory for signs of a cause.
#

###############################################################################
#
# License Acceptance
#
# Valid Options :
#       true   Accepts the license. Will install the product.
#       false  Declines the license. Install will not occur.
#
# If no install occurs, this will be logged to a temporary log file
# in the /tmp/InstallShield/plglogs temporary directory.
#
# By changing this value in this response file to "true", you agree that you
# have reviewed and agree to the terms of the IBM International Program License
```

```
# Agreement accompanying this program, which is located at <CD_ROOT>/plugin/lafiles.
# If you do not agree to these terms, do not change the value or otherwise download,
# install, copy, access, or use the program and promptly return the program and proof
# of entitlement to the party from whom you acquired it to obtain a refund of the
# amount you paid.
#

-OPT silentInstallLicenseAcceptance="false"


###############################################################################
#
# Operating System Prerequisite Checking
#
# If you want to disable operating system prerequisite checking, uncomment
# the following line. This will notify the installer to continue with
# the installation and log the warnings even though the prerequisite checking
# has failed.
#
#-OPT disableOSPrereqChecking="true"


###############################################################################
#
# install Type
#
# Valid Options :
#       local   WebSphere Application Server and Web server on same machine
#       remote  WebSphere Application Server and Web server on separate machines.
#

-OPT installType="local"


###############################################################################
#
# Plugins Install Location
#
# The desired install location of Web Server Plugins for IBM WebSphere
# Application Server V6.1. Specify a valid directory into which the product
# should be installed.  This directory must be either empty or not exist.
#
# Below is the default Web Server Plugins install location for the i5
# operating system.
#
# i5OS Default Install Location:
#
#  -OPT installLocation="/QIBM/ProdData/WebSphere/Plugins/V61/webserver"
#

-OPT installLocation="/QIBM/ProdData/WebSphere/Plugins/V61/webserver"


###############################################################################
#
# Profile Location
#
# The desired install location of Web Server Plugins profiles.  For new installs,
# specify a valid directory into which the profiles should be installed.  This
# directory must be either empty or not exist.
#
# Below is the default profile install location for the i5 operating system.
#
# i5OS Default Profile Install Location:
#
#  -OPT defaultProfileLocation="/QIBM/UserData/WebSphere/Plugins/V61/webserver"
#
```

```
-OPT defaultProfileLocation="/QIBM/UserData/WebSphere/Plugins/V61/webserver"


###############################################################################
#
# Allow Profile Location Override
#
# This option allows users to overrule the empty default profile location requirement.
#
# Valid values:
# true - Allow profile location override
# false - Do not allow profile location override
#
# If desired, uncomment the following entry
#
# -OPT allowOverrideProfileLocation="true"
#


###############################################################################
#
# WAS V6.1 Existing Location
#
# Valid Options : Existing WebSphere Application Server Version 6.1 install home directory.
#
# Note : This option is valid for local install type.
#        The install will use the directory entered below.
#
# Below is the default WebSphere Application Server install location for the i5
# operating system.
#
# i5OS Default Install Location:
#
#  -OPT wasExistingLocation="/QIBM/ProdData/WebSphere/AppServer/V61/<EDITION>"
#
#  where Edition = "Express" for WebSphere Application Server - Express
#                = "Base" for WebSphere Application Server
#                = "ND" for WebSphere Application Server Network Deployment
#

-OPT wasExistingLocation="/QIBM/ProdData/WebSphere/AppServer/V61/Express"


###############################################################################
#
# Web server to configure
#
# Valid options
#  : none          Install binaries only.  No Web server configuration.
#  : ihs           IBM HTTP Server V6 or V6.1
#  : domino6       Lotus Domino Web Server V6 or V6.5 (not supported on HP-UX)
#
# Note  : Specify only one Web server to configure.
#

-OPT webServerSelected="none"


#==============================================================================
#
# IHS-specific Administrator settings
#
#==============================================================================
#
# HTTP Administration Port number
#
```

```
# Specify the HTTP Administration Port number (only specify value when configuring IHS)
#

-OPT ihsAdminPort=""


#
# IHS Administrator User ID/Group for IHS Administration server
#
# Specify the IHS Administrator server userid and user group. (only specify value
# when configuring IHS)
#

-OPT ihsAdminUserID=""
-OPT ihsAdminUserGroup=""

#
#================================================================================


################################################################################
#
# Web Server Configuration File 1
#
# Valid options for Web Server configuration file 1 location
#
#  ihs   : httpd.conf
#  domino6  : Notes.jar
#
#  Note : File must exist
#
# Below is the default Web Server Configuration file location for the i5
# operating system.
#
# i5OS Default Web Server Configuration file Location:
#
#  For IHS (IBM HTTP Server)
#    -OPT webServerConfigFile1="/www/<YourWebServerName>/conf/httpd.conf"
#
#  For Lotus Domino 6
#    -OPT webServerConfigFile1="/QIBM/ProdData/LOTUS/DOMINO605/Notes.jar"
#

-OPT webServerConfigFile1=""


#================================================================================
#
# DOMINO-specific settings
#
#================================================================================
#
# Web server Configuration File 2
#
# Valid options for Web server configuration file 2 location
#
#  domino6 : names.nsf
#
# Note : File must exist
#
# File can be found under the DOMINO instance directory you defined when
# you created your DOMINO instance
#

-OPT webServerConfigFile2=""
```

```
###############################################################################
#
# Domino 6 User ID
#
# Specify the Domino 6 User ID.
#

-OPT domino6UserID="notes"

#
#==============================================================================


###############################################################################
#
# Web server port number
#
# Specify the Web server port for the Web server selected to be configured.
#

-OPT webServerPortNumber="80"


###############################################################################
#
# Web server Definition Name
#
# A web server definition allows for Web server administration through the WebSphere
# admin console.
#
#  Note : No spaces are allowed in the Web server definition name.
#

-OPT webServerDefinition="webserver1"


###############################################################################
#
# plugin-cfg.xml File Location
#
# This file will be generated by the plugin installer.
#
# Valid options:
#       "" : leaving the string empty will result in the installer generating the
#            plugin-cfg.xml file location at install time and configuring the Web
#            Server to use this location.  This is the recommended option.
#
#       "<file_location>" : User may enter an existing file location.  The Web Server
#                           will be configured to use this existing plugin-cfg.xml file
#                           location.  If a file is specified, it must exist, otherwise
#                           the install will not proceed.
#

-OPT pluginCfgXmlLocation=""


###############################################################################
#
# WebSphere Application Server Machine HostName
#
# remote install type  : enter the hostname of the WebSphere Application Server machine.
# local install type  : "" (hostname of the machine being installed to will be used.)
#

-OPT wasMachineHostName=""
```

```
###############################################################################
#
#  Advanced User Options available in silent installs only
#
#  Map all the existing deployed applications to the Web server.
#
#  Valid Options
#        true  :  Web server Definition is mapped as a target to the existing
#                 deployed applications such as snoop and hitcount (Recommended)
#        false :  No applications are mapped to the Web server definition.
#
#  Note : If not set to a valid option of true or false, the installer will set to
#         true and continue the install.
#

-OPT mapWebserverToApplications="true"


###############################################################################
#
#  Web server Hostname
#
#  In advanced scenarios where a user has multiple Web server hostnames on a
#  machine, set the entry below to the Web server hostname used to configure.
#
#  Valid Options :
#        "" :  Install will resolve to the hostname detected on the machine (Recommended)
#        "<HOSTNAME>" :  Where <HOSTNAME> is a Web server hostname on the machine.
#

-OPT webServerHostName=""


###############################################################################
#
# WAS Profile Name
#
# Specify the name of the WAS Profile to be configured. This option is only valid
# in local install scenarios.
#
# Valid options:
#       "<WAS_profile_name>" : User must enter the name of an existing WAS profile
#                              leaving the string empty will result in the installer
#                              using the name of the default profile
#                              Example:  -OPT profileName="AppSrv01"
#

-OPT profileName=""
```

Editing Web server configuration files

Use the IBM Web Administration for iSeries to configure i5/OS Web servers to communicate with a WebSphere Application Server. Doing so automatically updates the Web server configuration files as needed. See the *Installing your application serving environment* PDF for more information.

If you must change a configuration for some reason, you can use the IBM Web Administration for iSeries to reconfigure the Web server, or you can edit the files.The recommended approach is to always use the IBM Web Administration for iSeries to configure the Web server configuration file. However, sometimes you must edit the files.

This task shows you how to edit the Web server configuration files. Select one of the following links that identifies the Web server that you must reconfigure.

See the *Installing your application serving environment* PDF for information about configuring your Web server.

You can use the IBM Web Administration for iSeries to automatically configure IBM HTTP Server (powered by Apache) and Domino servers. You can also configure a Web server by editing its configuration.

Configuring Apache HTTP Server V2.0

This topic describes how to change configuration settings for Apache HTTP Server Version 2.0.

Important: If you are running IBM HTTP Server (powered by Apache) on i5/OS, you can use the manual configuration steps outlined below. However, it is recommended that you use the IBM Web Administration for i5/OS GUI. See the *Installing your application serving environment* PDF for more information.

Apache HTTP Server v2.0 is different from IBM HTTP Server (powered by Apache). Apache HTTP Server is not supported on i5/OS. For details on configuring IBM HTTP Server (Powered by Apache), see "Configuring IBM HTTP Server powered by Apache 2.0" on page 75.

When you install the Web server plug-ins for WebSphere Application Server, as described in "Installing Web server plug-ins" on page 9, the Plug-ins installation wizard configures the Web server. This topic describes how to configure the Apache HTTP Server Version 2.0 Web server. Other procedures in "Editing Web server configuration files" on page 72 describe configuring other supported Web servers.

Perform the step that configures Apache 2.0 for your operating system.

Examples and messages are shown on more than one line for clarity. Place each directive in a Web server configuration file on one line.

Local file path means a file path to the plugin-cfg.xml file on an Application Server that is on the same machine as the Web server. Remote file path means the file path to the plugin-cfg.xml file when the Application Server is on a remote machine. The Plug-ins installation wizard installs a dummy plugin-cfg.xml file during installation, but this file requires periodic propagation from the real file on the Application Server machine.

The *node_name* in the following Application Server local file paths is *web_server_name*_node for a stand-alone Application Server or *managed_node_name* for a managed node.

The name of the Web server definition in the following steps is webserver1.

Configure entries in thehttpd.conf file. It is recommended that you use the IBM Web Administration for i5/OS GUI to configure the httpd.conf file.

Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

```
LoadModule was_ap20_module /QSYS.LIB/QWAS6.LIB/QSVTAP20.SRVPGM
```

Local distributed example (Network Deployment only - the Web server is configured in a managed node:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V6/
   ND/profiles/profile1/config/cells/my_cell/nodes/
   my_managednode/servers/webserver1/plugin-cfg.xml
```

Local stand-alone example:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V6/
   ND/profiles/profile1/config/cells/my_cell/nodes/
   webserver1_node/servers/webserver1/plugin-cfg.xml
```

Remote example:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V6/
   ND/profiles/httpprofile1/config/webserver1/plugin-cfg.xml
```

This procedure results in reconfiguring the Apache 2.0 Web server.

If the IBM HTTP Server 1.3.2x directive, LoadModule ibm_app_server_http_module, is present in an IBM HTTP Server 2.0 httpd.conf file, the IBM HTTP Server cannot start. You must comment or delete the directive to start the Version 2 server.

The mod_was_ap20_http plug-in module requires the GSKIT Secure Sockets Layer (SSL) encryption library if the plug-in is configured to support encrypted connections to back-end WebSphere Application Servers.

The Plug-ins installation wizard installs the GSKIT SSL encryption library at the required level if it is not installed. If you manually copy the plug-in to a new machine, you might not have the required GSKIT libraries for encrypting back-end connections.

After configuring a Web server, you can install applications on it. See the Applications section of the information center for more information.

Configuring Lotus Domino

This task describes how to change configuration settings for Lotus Domino.

When you install the Web server plug-ins for WebSphere Application Server on a non-iSeries system, as described in “Installing Web server plug-ins” on page 9, the Plug-ins installation wizard configures the Web server.

This topic describes how to manually configure the Domino V6 Web server.

If you are running Domino Web Server on i5/OS, use the IBM Web Administration for iSeries GUI. See Creating and configuring an HTTP server instance for more information.

Other procedures in “Editing Web server configuration files” on page 72 describe configuring other supported Web servers.

Use the following procedure to enable the Web server plug-in to work with Lotus Domino.

1. Start the Domino server.
2. Access the names.nsf file using your Web browser (for example, http://tarheels2.raleigh.ibm.com/names.nsf). The browser prompts you for a password.
3. Give the administrator name and password.
4. Click **Configuration** on the left side of the page.
5. Click **Servers** on the left side of the page.
6. Click **All Server Documents** on the left side of the page.
7. Click the server that you intend to have work with WebSphere Application Server, Version 6.
8. Click **Edit Server** on the top-left of the center window.
9. Click **Internet Protocols** in the middle of the page.
10. Add the path to the Domino plug-in under the DSAPI Section in the middle-right of the page.

    The DSAPI filter location on i5/OS is /QSYS.LIB/QWAS61.LIB/LIBDOMINOH.SRVPGM. The plug-in is installed in the bin directory of the Web server plug-ins for WebSphere Application Server.

    If specifications for filter files for the Domino Server Application Programming Interface (DSAPI) already exist, use a space to delimit the Web server plug-in for WebSphere Application Server.

11. Click **Save and Close** in the upper-left of the center window.
12. Define the location of the plugin-cfg.xml configuration file.

    The location varies depending on how you have configured your system. If the Web server and the Application Server are on separate machines, you have a remote installation.

    If the two servers are on the same machine, you have a local installation.

    If the two servers are on the same machine and the Application Server node or the custom node is federated, you have a local distributed installation.

    In the following examples, webserver1 is the Web server definition name. **Setting the path to the plug-in configuration file**

    Edit your notes.ini file.

    a. From a CL command prompt, enter the Work with Domino Servers (WRKDOMSVR) command.
    b. Type **13** next to the applicable Domino server, then press **Enter**.
    c. Add or edit the WebSphereInit property. (See the following table.)

       To add a new line, type ″**i**″ next to the desired insertion line, then press **Enter**.
    d. Press **F3** to save and exit.

If the type of installation is:	Then use this command to set the environment variable:
Remote	`WebSpherePluginConfig` *`profile_root`*`/config/webserver1/plugin-cfg.xml`
Local stand-alone	`WebSpherePluginConfig` *`profile_root`*`/config/cells/my_cell/nodes/webserver1_node/servers/webserver1/plugin-cfg.xml`
Local distributed	`WebSpherePluginConfig` *`profile_root`*`/config/cells/my_cell/nodes/my_managednode/servers/webserver1/plugin-cfg.xml`

    During the installation process, the Plug-ins installation wizard creates the setupPluginCfg.sh file in two places:

    - The *plugins_root*/bin directory
    - The *lotus_root*/notesdata directory

    You can run the script from either location to set the WAS_PLUGIN_CONFIG_FILE environment variable. However, if you are reconfiguring the Web server, you might want to set the path yourself by setting the value of the environment variable with a path from the preceding table.

    The setupPluginCfg.sh script sets the file path value to the file path that the wizard configured originally. If you are reconfiguring the Web server to change the original file path, do not use this script.
13. Restart the Domino server. When the server starts, information similar to the following example is displayed:

```
01/21/2005 01:21:51 PM  JVM: Java virtual machine initialized
WebSphere Application Server DSAPI filter loaded
01/21/2005 01:21:52 PM  HTTP Web Server started
```

This procedure results in reconfiguring Version 6.x of Lotus Domino.

After configuring a Web server, you can install applications on it. See the Applications section of the information center for more information.

For more information on configuring Lotus Domino to work with WebSphere Application Server, search the Lotus Support Services Web site at http://www-3.ibm.com/software/lotus/support/. Enter the search term WebSphere in the keyword search field.

Configuring IBM HTTP Server powered by Apache 2.0

This topic describes how to change configuration settings for IBM HTTP Server powered by Apache 2.0.

If you are running IBM HTTP Server (powered by Apache) on i5/OS, you can use the manual configuration steps outlined below with the IBM Web Administration for iSeries. See Creating and configuring an HTTP server instance for more information.

When you install the Web server plug-ins for WebSphere Application Server, as described in “Installing Web server plug-ins” on page 9, the Plug-ins installation wizard configures the Web server on distributed platforms. This topic describes how to configure the IBM HTTP Server, V2.x. Other procedures in “Editing Web server configuration files” on page 72 describe configuring other supported Web servers.

Perform the step that configures IBM HTTP Server 6.0 version for your operating system.

Examples and messages are shown on more than one line for clarity. Place each directive in a Web server configuration file on one line.

Local file path means a file path to the plugin-cfg.xml file on an Application Server that is on the same machine as the Web server. Remote file path means the file path to the plugin-cfg.xml file when the Application Server is on a remote machine. The Plug-ins installation wizard installs a dummy plugin-cfg.xml file during installation, but this file requires periodic propagation from the real file on the Application Server machine.

The Web server definition name in the following examples is webserver1.

The node name *node_name* in the following Application Server local file paths is *web_server_name*_node for a stand-alone Application Server or *managed_node_name* for a managed node.

Configure entries in the httpd.conf file.

Use the IBM Web Administration for iSeries GUI to configure the httpd.conf file.

Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

```
LoadModule was_ap20_module
   /QSYS.LIB/QWAS61.LIB/QSVTAP20.SRVPGM
```

Local distributed example (Web server is configured in a managed node):

```
WebSpherePluginConfig
   profile_root/config/cells/my_cell/
   nodes/my_managednode/servers/webserver1/plugin-cfg.xml
```

Example:

```
WebSpherePluginConfig
   profile_root/
   config/webserver1/plugin-cfg.xml
```

This procedure results in editing and reconfiguring IBM HTTP Server powered by Apache 2.0.

If the IBM HTTP Server 1.3.2x directive, LoadModule ibm_app_server_http_module, is present in an IBM HTTP Server 2.0 httpd.conf file, the IBM HTTP Server cannot start. You must comment or delete the directive to start the Version 2 server.

The mod_was_ap20_http plug-in module requires the GSKIT Secure Sockets Layer (SSL) encryption library if the plug-in is configured to support encrypted connections to back-end WebSphere Application Servers.

The Plug-ins installation wizard installs the GSKIT SSL encryption library at the required level if it is not installed. If you manually copy the plug-in to a new machine, you might not have the required GSKIT libraries for encrypting back-end connections.

After configuring a Web server, you can install applications on it. See the Applications section of the information center for more information.

Configuring IBM HTTP Server Version

This topic describes how to change configuration settings for IBM HTTP Server.

IBM HTTP Server version 6.x is not supported on i5/OS. See the *Installing your application serving environment* PDF for information on how to configure IBM HTTP Server powered by Apache 2.0.

When you install the Web server plug-ins for WebSphere Application Server, as described in “Installing Web server plug-ins” on page 9, the Plug-ins installation wizard configures the Web server. This topic describes how to configure IBM HTTP Server, V6.x. Other procedures in “Editing Web server configuration files” on page 72 describe configuring other supported Web servers.

Perform the step that configures IBM HTTP Server for your operating system.

Examples and messages are shown on more than one line for clarity. Place each directive in a Web server configuration file on one line.

Local file path means a file path to the plugin-cfg.xml file on an Application Server that is on the same machine as the Web server. Remote file path means the file path to the plugin-cfg.xml file when the Application Server is on a remote machine. The Plug-ins installation wizard installs a dummy plugin-cfg.xml file during installation, but this file requires periodic propagation from the real file on the Application Server machine.

The *node_name* in the following Application Server local file paths is *web_server_name*_node for a stand-alone Application Server or *managed_node_name* for a managed node.

This procedure results in editing and reconfiguring IBM HTTP Server.

If the IBM HTTP Server V1.3.x directive, LoadModule ibm_app_server_http_module, is present in an IBM HTTP Server V6.x and later httpd.conf file, IBM HTTP Server cannot start. You must comment or delete the directive to start the Version 6.x server.

The mod_was_ap20_http plug-in module requires the GSKIT Secure Sockets Layer (SSL) encryption library if the plug-in is configured to support encrypted connections to back-end WebSphere Application Servers.

The Plug-ins installation wizard installs the GSKIT SSL encryption library at the required level if it is not installed. If you manually copy the plug-in to a new machine, you might not have the required GSKIT libraries for encrypting back-end connections.

After configuring a Web server, you can install applications on it. See the Applications section of the information center for more information.

Uninstalling the Web server plug-ins for WebSphere Application Server

On i5/OS, you cannot uninstall the Web server plugins without installing the WebSphere Application Server product. Use the **removeOs400WebServerDefinition** command to delete a Web server definition.

For example, to remove the definition for the Web server WEBSERVER1, which is associated with a profile called myHttpProfile, run this command from Qshell:

```
app_server_root/bin/removeOs400WebserverDefinition
   -webserver.name WEBSERVER1
   -profileName myHttpProfile
```

This topic describes uninstalling the Web server plug-ins for WebSphere Application Server.

If you used the IBM Key Management wizard to create SSL key files in the Web server Plug-ins home directory, back up the files to a directory outside of the Web server Plugins directory. After the uninstall procedure is complete, you can delete the SSL key files if they are no longer required.

After uninstalling the Web server plug-ins for WebSphere Application Server, you can reinstall them using any of the installation procedures described in “Installing Web server plug-ins” on page 9

1. Stop the Web server to allow the uninstaller program to change the Web server configuration.
2. Open a command window.
3. Change directories to the *plugins_root*/uninstall directory and issue the uninstall -silent command. **OR:** Change directories to the *plugins_root*/bin directory and issue the uninstall command.
4. Wait until the uninstall completes.

   Uninstalling the Web server plug-ins for WebSphere Application Server removes many files in the installation root directory, but leaves the following logs in the *plugins_root*/log/uninstall directory:
   - ISMP uninstall log: log.txt
   - Configuration uninstall log: masterConfigurationLog.txt
   - Web server deconfiguration log: uninstall*web_server_name*Plugin.log, such as the uninstallApachePlugin.log.
5. Delete files from the system temporary directory.

   Delete the following files:
   - Install temporary log : temporaryPluginInstallLog.txt
   - Uninstall temporary log : temporaryPluginUnInstallLog.txt
6. Delete the Web server definition in a stand-alone application server.

   The uninstaller program for the Web server plug-ins for WebSphere Application Server does not delete Web server definitions. However, you can delete a Web server definition from admin console OR by using the following wsadmin commands:

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

After you exit from the *plugins_root*/uninstPlugin directory, the directory is removed.

The only remaining directory is the *plugins_root*/logs directory. The logs directory contains the install process log, the uninstall process log, and the Web server creation logs.

Important: The only other files that might exist are the SSL key files that you can create using the IBM Key Management Wizard. You can move these files to a safe location before using the manual uninstalling procedure, if necessary.

To reinstall the Web server plug-ins for WebSphere Application Server, launch the installation procedure again.

To reinstall the Web server plug-ins for WebSphere Application Server into the original directory, delete the existing installation root directory for the plug-ins before reinstalling.

The default location is shown in “Directory conventions,” on page 471.

See “Installing Web server plug-ins” on page 9 for information about installation scenarios for reinstalling Web server plug-ins.

See “Manually uninstalling Web server plug-ins for WebSphere Application Server” for information about manually uninstalling Web server plug-ins.

Manually uninstalling Web server plug-ins for WebSphere Application Server

You can use this manual procedure for uninstalling Web server plug-ins for WebSphere Application Server when the uninstaller program is not available for some reason.

Important: If you are running WebSphere Application Server on a i5/OS platform, see “Uninstalling the Web server plug-ins for WebSphere Application Server” on page 77.

This topic describes uninstalling the Web server plug-ins for WebSphere Application Server.

After uninstalling the Web server plug-ins for WebSphere Application Server, you can reinstall them using any of the installation procedures that are described in “Installing Web server plug-ins” on page 9

1. Delete the installation root directory for Web server plug-ins for WebSphere Application Server.
2. Delete the Web server definition in the application server configuration.

   You can delete a Web server definition from the admin console OR by using the following wsadmin commands:

   ```
   $AdminTask deleteServer { -serverName   webserver1 -nodeName WebserverHostName-node_node }
   $AdminTask removeUnmanagedNode { -nodeName WebserverHostName-node_node }
   $AdminConfig save
   ```
3. Reconfigure any Web servers that were configured to use the binary plug-ins that you deleted.

   See “Editing Web server configuration files” on page 72.

Deleting the installation root directory for the Web server plug-ins for WebSphere Application Server removes the binary plug-ins. Any Web servers that are configured to use the deleted binary modules do not work. Reinstall the Web server plug-ins for WebSphere Application Server and reconfigure the Web servers to restore their functionality.

See “Installing Web server plug-ins” on page 9 for information about other installation scenarios for reinstalling Web server plug-ins.

Allowing Web servers to access the administrative console

This topic describes how to add the virtual host that servers the administrative console to the plug-in configuration file so that you can access the administrative console through a Web server.

Install your Version 6 WebSphere Application Server product, a Web server, and the Web server plug-ins for WebSphere Application Server.

When you configure a Web server plug-in, a Web server definition is created on the Application Server system, either directly when they are on the same machine, or by a script for remote scenarios.

After creating the Web server definition, the plug-in configuration file exists within the Web server definition.

The plugin-cfg.xml file can be overwritten by the deployment manager sync operation, the GenPluginCfg script or any other method that regenerates the file. If you make changes to the plugin-cfg.xml file, and want to keep those changes, it is recommended that you create a copy of the file in a separate location. Make your manual updates each time the file is automatically refreshed by another process.

This task gives you the option of configuring the admin_host so that Web servers can access the administrative console. When the Web server plug-in configuration file is generated, it does not include admin_host on the list of virtual hosts.

1. Use the administrative console to change the admin_host virtual host group to include the Web server port (80 by default).
   a. Click **Environment > Virtual Host > admin_host > Host Aliases > New**.

      The default port that displays is 80, unless you specify a different port during profile creation.
   b. Specify the IP address, or the name of the machine that is hosting the HTTP server.

      For example, if you installed a WebSphere Application Server product on a machine that is named waslwaj.rtp.ibm.com, specify the name in this field.
2. Click **Apply > Save**.
3. Stop and restart the application server.

   For example, to access the administrative console of a stand-alone application server, stop and restart the server1 process.

   Start a Qshell session and run the following command:

   ```
   cd profile_root/bin
   stopServer server1
   ```

   Then issue the following command to stop the application server:

   ```
   stopServer -profileName myProfile server1
   ```

   After receiving the following message, you can restart the application server:

   ```
   Server server1 stop completed.
   ```

   To start the application server, issue the following command:

   ```
   startServer server1
   ```

   When the application server is running, a message is displayed that indicates that the process is running. This message includes the iSeries job ID and the administrative console port.
4. Stop and restart a deployment manager.

   For example, to access the administrative console of a deployment manager, stop and restart the deployment manager.

   Start a Qshell session and run the following command:

   cd *profile_root*/bin

   Then issue the following command to stop the deployment manager:

   ```
   stopManager
   ```

   After receiving the following message, you can restart the deployment manager:

   ```
   Server dmgr stop completed.
   ```

   To start the deployment manager, issue the following command:

   ```
   startManager
   ```

   When the deployment manager is running, a message is displayed that indicates that the process is running. This message includes the iSeries job ID and the administrative console port.
5. Edit the plugin-cfg.xml file to include the following entries:

   ```
   <VirtualHostGroup Name="admin_host">
           <VirtualHost Name="*:13060"/>
      </VirtualHostGroup>
       ...
       ...
       ...
       <ServerCluster Name="my60Profile.dmgr_muiSeries_Cluster">
           <Server LoadBalanceWeight="1" Name="myiSeries_my60Profile.dmgr">
               <Transport Hostname="myiSeries" Port="11060" Protocol="http"/>
           </Server>
   ```

```
    <PrimaryServers>
        <Server Name="myiSeries_my60Profile.dmgr"/>
    </PrimaryServers>
</ServerCluster>
...
...
...
<UriGroup Name="admin_host_my60Profile.dmgr_myiSeries_Cluster_URIs">
    <Uri AffinityCookie="JSESSIONID"
        AffinityURLIdentifier="jsessionid" Name="/ibm/console/*"/>
</UriGroup>
<Route ServerCluster="my60Profile.dmgr_myiSeries_Cluster"
    UriGroup="admin_host_my60Profile.dmgr_myiSeries_Cluster_URIs" VirtualHostGroup="admin_host"/>
```

If your HTTP server has an HTTP port other than 80 please add an entry to the VirtualHostGroup:

```
<VirtualHost Name="*:port"/>
```

The *port* variable is your HTTP server port.

You can configure your supported Web servers to access the administrative console application of a deployment manager or a stand-alone application server.

Web server plug-in properties settings

Use this page to view or change the settings of a Web server plug-in configuration file. The plug-in configuration file, plugin_cfg.xml, provides properties for establishing communication between the Web server and the Application Server.

To view this administrative console page, click **Servers > Web Servers >** *web_server_name* **Plug-in Properties**.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information.

The **Runtime** tab is available only when this Web server has accessed applications running on application servers and there is an http_plugin.log file.

Ignore DNS failures during Web server startup

Specifies whether the plug-in ignores DNS failures within a configuration when starting.

This field corresponds to the IgnoreDNSFailures element in the plugin-cfg.xml file.

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

Data type	String
Default	false

Refresh configuration interval

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

In a development environment in which changes are frequent, a lower setting than the default setting of 60 seconds is preferable. In production, a higher value than the default is preferable because updates to the

configuration will not occur so often. If the plug-in reload fails for some reason, a message is written to the plug-in log file and the previous configuration is used until the plug-in configuration file successfully reloads. If you are not seeing the changes you made to your plug-in configuration, check the plug-in log file for indications of the problem.

Data type	Integer
Default	60 seconds.

Plug-in configuration file name

Specifies the file name of the configuration file for the plug-in. The Application Server generates the plugin-cfg.xml file by default. The configuration file identifies applications, Application Servers, clusters, and HTTP ports for the Web server. The Web server uses the file to access deployed applications on various Application Servers.

You can change the name of the plug-in configuration file. However, if you do change the file name, you must also change the Web server configuration to point to the new plug-in configuration file.

If you select a Web server plug-in during installation, the installer program configures the Web server to identify the location of the plugin-cfg.xml file, if possible. The plug-in configuration file, by default, is installed in the *plugins_root*/config/*web_server_name* directory.

The installer program adds a directive to the Web server configuration that specifies the location of the plugin-cfg.xml file.

For remote Web servers, you must copy the file from the local directory where the Application Server is installed to the remote machine. This is known as propagating the plug-in configuration file. If you are using an IBM® HTTP Server V6.1 for your Web server, WebSphere® Application Server can automatically propagate the plug-in configuration file for you to remote machines provided there is a working HTTP transport mechanism to propagate the file.

You can click **View** to display a copy of the current plug-in configuration file.

Data type	String
Default	plugin-cfg.xml

Automatically generate plug-in configuration file

To automatically generate a plug-in configuration file to a remote Web server:

- This field must be checked.
- The plug-in configuration service must be enabled

When the plug-in configuration service is enabled, a plug-in configuration file is automatically generated for a Web server whenever:

- The WebSphere Application Server administrator defines new Web server.
- An application is deployed to an Application Server.
- An application is uninstalled.
- A virtual host definition is updated and saved.

By default, this field is checked. Clear the check box if you want to manually generate a plug-in configuration file for this Web server.

Automatically propagate plug-in configuration file

Specifies whether or not you want the application server to automatically propagate a copy of a changed plug-in configuration file to a Web server:

- This field must be checked.
- The plug-in configuration service must be enabled
- A WebSphere Application Server node agent must be on the node that hosts the Web server associated with the changed plug-in configuration file.

By default, this field is checked.

Note: The plug-in configuration file can only be automatically propagated to a remote Web server if that Web server is an IBM HTTP Server V6.1 Web server and its administration server is running.

Because the plug-in configuration service runs in the background and is not tied to the administrative console, the administrative console cannot show the results of the automatic propagation.

Plug-in key store file name

Specifies the fully qualified directory path and file name of the database file containing your security key rings that the Web server plug-in uses for HTTPS requests. This file resides on the Web server that is associated with this Web server plug-in. After you specify the fully qualified directory path and file name of the database file, you can:

- Cick **Manage keys and certificates** to update this file.
- Click **Copy to Web server key store directory** to add a copy of this file to the key store directory for the Web server.

Data type	String
Default	None

Plug-in configuration directory and file name

Specifies the fully qualified path of the Web server copy of the Web server plug-in configuration file. This path is the name of the file and its location on the machine where the Web server is running.

Plug-in key store directory and file name

Specifies the fully qualified path of the Web server copy of the database file that contains your security key rings. This path is the name of the file and its location on the machine where the Web server is running.

Plug-in logging

Specifies the location and name of the http_plugin.log file. Also specifies the scope of messages in the log.

This field corresponds to the RequestMetrics traceLevel element in the plugin-cfg.xml file.

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

On a distributed platform, if the log file does not exist then it will be created. If the log file already exists, it will be opened in append mode and the previous plug-in log messages will remain.

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

Data type	String

Default *plugins_root*/logs/*web_server_name*/http_plugin.log

Specify the file path of the http_plugin.log file.

Log level- The level of detail of the log messages that the plug-in should write to the log. You can specify one of the following values for this attribute:
- Trace. All of the steps in the request process are logged in detail.
- Stats. The server selected for each request and other load balancing information relating to request handling is logged.
- Warn. All warning and error messages resulting from abnormal request processing are logged.
- Error. Only error messages resulting from abnormal request processing are logged.
- Debug. All of the critical steps performed in processing requests are logged.
- Detail. All of the information about requests and responses are logged.

If a Log level is not specified, the default value **Error** is used.

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

Data type String
Default Error

Web server plug-in request and response optimization properties settings

Use this page to view or change the request and response optimization properties for a Web server plug-in.

To view this administrative console page, click **Servers > Web Servers >** *web_server_name* **Plug-in Properties > Request and Response**.

Maximum chunk size used when reading the HTTP response body

Specifies the maximum chunk size the plug-in can use when reading the response body.

This field corresponds to the ResponseChunkSize element in the plugin-cfg.xml file.

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

If the content length of the response body is unknown, the values specified for this property is used as the size of the buffer that is allocated. The response body is then read in this size chunks, until the entire body is read. If the content length is known, then a buffer size of either the content length or the specified size (whichever is less) is used to read the response body.

Data type Integer
Default 64 kilobytes

Specify the size in kilobytes (1024 byte blocks).

Enable Nagle algorithm for connections to the Application Server

When checked, the Nagle algorithm is enabled for connections between the plug-in and the Application Server.

This field corresponds to the ASDisableNagle element in the plugin-cfg.xml file.

The Nagle algorithm is named after engineer John Nagle, who invented this standard part of the transmission control protocol/internet protocol (TCP/IP). The algorithm reduces network overhead by adding a transmission delay (usually 20 milliseconds) to a small packet, which lets other small packets arrive and be included in the transmission. Because communications has an associated cost that is not as dependent on packet size as it is on frequency of transmission, this algorithm potentially reduces overhead with a more efficient number of transmissions.

By default, this field is not checked, and the Nagle algorithm is disabled. Select this field to enable the Nagle algorithm.

Enable Nagle Algorithm for the IIS Web Server

When checked, the Nagle algorithm is used for connections from the Microsoft Internet Informations Services (IIS) Web Server to the Application Server.

This field corresponds to the IHSDisableNagle element in the plugin-cfg.xml file. It only appears if you are using the Microsoft Internet Informations Services (IIS) Web server.

By default, this field is not checked, and the Nagle algorithm is disabled. Select this field to enable the Nagle algorithm for this connection.

Chunk HTTP response to the client

When checked, responses to the client are broken into chunks if a `Transfer-Encoding : Chunked` response header is present in the response.

This field corresponds to the ChunkedResponse element in the plugin-cfg.xml file. It only appears if you are using a Microsoft Internet Informations Services (IIS) Web Server, a Java System Web server, or a Domino Web server. The IBM HTTP Server automatically handles breaking the response into chunks to send to the client.

By default, this field is not checked, and responses are not broken into chunks. Select this field to enable responses to the client to be broken into chunks if a `Transfer-Encoding : Chunked` response header is present in the response.

Accept content for all requests

This field corresponds to the AcceptAllContent element in the plugin-cfg.xml file.

When selected, users can include content in POST, PUT, GET, and HEAD requests when a Content-Length or Transfer-encoding header is contained in the request header.

By default, this field is not checked. Select this field to enable users to include content in POST, PUT, GET, and HEAD requests when a Content-Length or Transfer-encoding header is contained in the request header.

Virtual host matching

When selected, virtual host mapping is performed by physically using the port number for which the request was received.

This field corresponds to the VHostMatchingCompat element in the plugin-cfg.xml file.

By default, this field is not checked, and matching is done logically using the port number contained in the host header. Select this field if you want virtual host mapping performed by physically using the port number for which the request was received.

Use the radio buttons to make your physical or logical port selection.

Application server port preference
Specifies which port number the Application Server should use to build URI's for a sendRedirect.

This field corresponds to the AppServerPortPreference element in the plugin-cfg.xml file.

Specify:
- `webserverPort` if the port number from the host header of the HTTP request coming in is to be used.
- `hostHeader` if the port number on which the Web server received the request is to be used.

The default is `webserverPort`.

Web server plug-in caching properties settings
Use this page to view or change the caching properties for a Web server plug-in.

To view this administrative console page, click **Servers > Web Servers >** *Web_server_name* **Plug-in Properties > Caching Properties**.

Enable Edge Side Include (ESI) processing to cache the responses
Specifies whether to enable Edge Side Include processing to cache the responses.

This field corresponds to the esiEnable element in the plugin-cfg.xml file.

By default, this field is not checked. Select this field if you want Edge Side Include (ESI) processing used to cache responses. If ESI processing is disabled for the plug-in, the other ESI plug-in properties are ignored. Clear the checkbox to disable Edge Side Include processing.

Enable invalidation monitor to receive notifications
When checked, the ESI processor receives invalidations from the application server.

This field corresponds to the ESIInvalidationMonitor element in the plugin-cfg.xml file. It is ignored if Edge Side Include (ESI) processing is not enabled for the plug-in.

By default, this field is selected. Clear the check box if you do not want the application server to send invalidations to the ESI processor.

Maximum cache size
Specifies, in 1K byte units, the maximum size of the cache. The default maximum size of the cache is 1024K bytes (1 megabyte). If the cache is full, the first entry to be evicted from the cache is the entry that is closest its expiration time.

This field corresponds to the esiMaxCacheSize element in the plugin-cfg.xml file.

Data type	Integer
Default	1024 kilobytes
	Specify the size in kilobytes (1024 byte blocks).

Web server plug-in request routing properties settings
Use this page to view or change the request routing properties for a Web server plug-in.

To view this administrative console page, click **Servers > Web Servers >** *Web_server_name* **Plug-in Properties > Plug-in** *server_cluster_name* **Properties**.

Load balancing option

Specifies the load balancing option that the plug-in uses in sending requests to the various application servers associated with that Web server.

This field corresponds to the LoadBalanceWeight element in the plugin-cfg.xml file.

Select the appropriate load balancing option:

- Round robin
- Random

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

The default load balancing type is Round Robin.

Retry interval

Specifies the length of time, in seconds, that should elapse from the time an application server is marked down to the time that the plug-in retries a connection.

This field corresponds to the ServerWaitforContinue element in the plugin-cfg.xml file.

Data type	Integer
Default	60 seconds

Maximum size of request content

Select whether there is a limit on the size of request content. If limited, this field also specifies the maximum number of kilobytes of data a request can contain. When a limit is set, the plug-in fails any request that is received that contains more data than the specified limit.

This field corresponds to the PostSizeLimit element in the plugin-cfg.xml file.

Select whether to limit the size of request content:

- `No limit`
- `Set limit`

If you select `Set limit`, specify a limit size.

Data type	Integer
Default	Specify the size in kilobytes (1024 byte blocks). `-1`, which indicates there is no limit for the post size.

Maximum buffer size used when reading HTTP request content

Specifies, in kilobytes, the maximum buffer size that is used when the content of an HTTP request is read. If the application server that initially receives a request cannot process that request, the data contained in this buffer is sent to another application server in an attempt to have that application serverprocess the request.

This field corresponds to the PostBufferSize element in the plugin-cfg.xml file.

If **Set limit** is selected, specify a limit size.

Data type	Integer
	Specify the size in kilobytes (1024 byte blocks).
Default	64

Remove special headers
When checked, the plug-in will remove any headers from incoming requests before adding the headers the plug-in is supposed to add before forwarding the request to an application server.

This field corresponds to the RemoveSpecialHeaders element in the plugin-cfg.xml file.

The plug-in adds special headers to the request before it is forwarded to the application server. These headers store information about the request that will need to be used by the application. Not removing the headers from incoming requests introduces a potential security exposure.

By default, the special headers are not retained. Clear the check box to retain special headers.

Clone separator change
When this option is selected, the plug-in expects the plus character (+) as the clone separator.

This field corresponds to the ServerCloneID element in the plugin-cfg.xml file.

Some pervasive devices cannot handle the colon character (:) used to separate clone IDs in conjunction with session affinity. If this field is checked, you must also change the configurations of the associated application servers such that the application servers separates clone IDs with the plus character as well.

By default, this option is selected. Clear the field if you want to use the colon character to separate clone IDs.

Web server plug-in configuration service property settings

Use this page to view or change the configuration settings for the Web server plug-in configuration service.

If you are using a stand-alone application server, click **Application Servers >** *server_name* > **Administration Services** > **Web server plug-in configuration service** to view this administrative console page.

If you are using the deployment manager, click **System Administration > Deployment manager > Administration Services > Web server plug-in configuration service**.

Enable automated Web server configuration processing

The Web server plug-in configuration service is selected by default. The service automatically generates the plug-in configuration file whenever the Web server environment changes, with a few exceptions. For example, the plug-in configuration file is regenerated whenever one of the following activities occurs:

- A new application is deployed on an associated application server
- The Web server definition is saved
- An application is removed from an associated application server
- A new virtual host is defined

The plug-in configuration file does not regenerate when:

- A cluster member is added to a cluster

- TCP channel settings are updated for an application server

By default, this option is selected. Clear the field to disable automated Web server configuration processing.

Application Server property settings for a Web server plug-in

Use this page to view or change application server settings for a Web server plug-in.

To view this administrative console page, click **Application Servers >** *server_name*, and then under Additional Properties, click **> Web server plug-in properties**.

Server role

Specifies the role this application server is assigned.

Select `Primary` to add this application server to the list of primary application servers. The plug-in initially attempts to route requests to the application servers on this list.

Select `Backup` to add this application server to the list of backup application servers. The plug-in does not load balance across the backup application servers. A backup server is only used if a primary server is not available. When the plug-in determines that a backup application server is required, it goes through the list of backup servers, in order, until no servers are left in the list or until a request is successfully sent and a response received from one of the servers on this list.

Default setting Primary

Read/Write timeout

Specifies whether there is a time limit for how long the plug-in waits to send a request to or receive a response from the application server. If `Set Timeout` is selected, you must specify the length of time, in seconds that the plug-in waits to send a request or to receive a response. When selecting a value to specify for this field, remember that it might take a couple of minutes for an application server to process a request. Setting the value too low might cause the plug-in to send a false server error response to the client.

If you select `No Timeout`, the plug-in uses blocked I/O to write requests to and read responses from the application server until the TCP connection times out.

This field is ignored for a plug-in running on a Solaris platform.

Data type Integer
Default No Timeout

Connect timeout

Specifies whether or not there is a limited amount of time the application server will maintain a connection with the Web server.

You can select either **No timeout** or **Set timeout**. If you select **Set timeout** you, must specify, in seconds, the length of time a connection with the Web server is to be maintained.

This property enables the plug-in to perform non-blocking connections with the application server. Non-blocking connections are beneficial when the plug-in is unable to contact the destination to determine whether or not the port is available. If no value is specified for this property, the plug-in performs a

blocking connect in which the plug-in sits until an operating system times out (which could be as long as 2 minutes depending on the platform) and allows the plug-in to mark the server `unavailable`.

A value of 0 causes the plug-in to perform a blocking connect. A value greater than 0 specifies the number of seconds you want the plug-in to wait for a successful connection. If a connection does not occur after that time interval, the plug-in marks the server unavailable and fails over to another application server defined for the requested application.

Data type	Integer
Default	0

Maximum number of connections that can be handled by the Application Server

Specifies the maximum number of pending connections to an Application Server that can be flowing through a Web server process at any point in time.

This field corresponds to the ServerMaxConnections element in the plugin-cfg.xml file.

You can select either **No limit** or **Set limit**. If you select **Set limit** you, must specify the maximum number of connections that can exist between the Web server and the Application Server at any given point in time.

For example, assuming that:

- The application server is fronted by 5 nodes that are running an IHS Web server.
- Each node starts 2 processes.
- This property is set to 50.

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

If this attribute is set to either zero or -1, there is no limit to the number of pending connections to the Application Servers.

Data type	Integer
Default	-1

Use extended handshake to check whether application server is running

When selected, the Web server plug-in will use an extended handshake to check whether or not the Application Server is running.

This field corresponds to the ServerExtendedHandshake element in the plugin-cfg.xml file.

Select this property if a proxy firewall is between the plug-in and the application server.

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

If the plug-in performs some handshaking with the application server to ensure that it is started before it sends a request it can failover to another application server if it detects that the application server with which it is attempting to perform a handshake is down.

By default, this field is not checked. Select this field if you want to use extended handshake to check whether an application server is running.

Send the header ″100 Continue″ before sending the request content

This field corresponds to the WaitForContinue element in the plugin-cfg.xml file.

When selected, the Web server plug-in will send the header ″100 Continue″ to the application server before it sends the request content.

By default, this field is not checked. Select this field to enable this function.

Web server plug-in configuration properties

The following table indicates which panel in the administrative console you need to use to manually configure a Web server plug-in property.

Table 12. Web server plug-in configuration properties

Administrative console panel	Field name	Configuration property name
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties**	Refresh configuration interval	RefreshInterval
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties**	Plug-in log file name	Log->name
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties**	Plug-in logging	Log->LogLevel
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties**	Ignore DNS failures during Web server startup	IgnoreDNSFailures
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties**	KeyringLocation	Keyring
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties**	StashfileLocation	Stashfile
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Custom properties > New**	FIPSEnable	FIPSEnable

Table 12. Web server plug-in configuration properties (continued)

In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request routing**	Load balancing option	LoadBalance
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request Routing**	Clone separator change	CloneSeparatorChange
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request Routing**	Retry interval	RetryInterval
In the administrative console, click **Servers >** *Web server_name* **> Plug-in properties > Request routing**	Maximum size of request content	PostSizeLimit
In the administrative console, click **Servers >** *Web server_name* **> Plug-in properties > Request routing**	Size of the buffer that is used to cache POST requests	PostBufferSize
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request routing**	Remove special headers	RemoveSpecialHeaders
In the administrative console, click **Application Servers >** *server_name* **> Web server plug-in properties**	Server role	PrimaryServers and BackupServers list
In the administrative console, click **Application Servers >** *server_name* **> Web server plug-in properties**	Connect timeout	Server ConnectTimeout
In the administrative console, click **Application Servers >** *server_name* **> Web server plug-in properties**	The read and write timeouts for all the connections to the application server	ServerIOTimeout
In the administrative console, click **Application Servers >** *server_name* **> Web server plug-in properties**	Use extended handshake to check whether Application Server is running	Server Extended Handshake
In the administrative console, click **Application Servers >** *server_name* **> Web server plug-in properties**	Send the header ″100 Continue″ before sending the request content	WaitForContinue
In the administrative console, click **Application Servers >** *server_name* **> Web server plug-in properties**	Maximum number of connections that can be handled by the Application Server	Server MaxConnections
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Application server port preference	AppServerPortPreference
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Enable Nagle algorithm for connections to the Application Server	ASDisableNagle

Table 12. Web server plug-in configuration properties (continued)

In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Enable Nagle Algorithm for the IIS Web Server	IISDisableNagle
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Virtual host matching	VHostMatchingCompat
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Maximum chunk size used when reading the response body	ResponseChunkSize
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Accept content for all requests	AcceptAllContent
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Chunk HTTP response to the client	ChunkedResponse
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Priority used by the IIS Web server when loading the plug-in configuration file	IISPluginPriority
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Caching**	Enable Edge Side Include (ESI) processing to cache the responses	ESIEnable
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Caching**	Maximum cache size	ESIMaxCacheSize
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Caching**	Enable invalidation monitor to receive notifications	ESIInvalidationMonitor

Web server plug-in connections

The WebSphere Application Server Web server plug-ins are used to establish and maintain persistent HTTP and HTTPS connections to Application Servers .

When the plug-in is ready to send a request to the application server, it first checks its connection pool for existing connections. If an existing connection is available the plug-in checks its connection status. If the status is still good, the plug-in uses that connection to send the request. If a connection does not exist, the plug-in creates one. If a connection exists but has been closed by the application server, the plug-in closes that connection and opens a new one.

After a connection is established between a plug-in and an application server, it will not be closed unless the application server closes it for one of the following reasons:

- If the **Use Keep-Alive** property is selected and the time limit specified on the **Read timeout** or **Write timeout** property for the HTTP inbound channel has expired.
- The maximum number of persistent requests which can be processed on an HTTP inbound channel has been exceeded. (This number is set using the HTTP inbound channel's **Maximum persistent requests** property.)
- The Application Server is shutting down.

Even if the application server closes a connection, the plug-in will not know that it has been closed until it tries to use it again. The connection will be closed if one of the following events occur:

- The plug-in receives a new HTTP request and tries to reuse the existing connection.
- The number of **httpd** processes drop because the Web server is not receiving any new HTTP requests. (For the IBM HTTP Server, the number of **httpd** processes that are kept alive depends on the value specified on the Web server's MinSpareServers directive.)
- The Web server is stopped and all **httpd** processes are terminated, and their corresponding sockets are closed.

Important: Sometimes, if a heavy request load is stopped or decreased abruptly on a particular application server, a lot of the plug-in's connections to that application server will be in CLOSE_WAIT state. Because these connections will be closed the first time the plug-in tries to reuse them, having a large number of connections in CLOSE-WAIT state should not affect performance

Web server plug-in remote user information processing

You can configure your Web server with a third-party authentication module and then configure the Web server plug-in to route requests to an application server.

If an application calls the getRemoteUser() method, it relies on a private HTTP header that contains the remote user information and is parsed by the plug-in. The plug-in sets the private HTTP header value whenever a Web server authentication module populates the remote user in the Web server data structure. If the private HTTP header value is not set, the application's call to getRemoteUser() returns a null value.

- In the case of an Apache Web server or the IBM HTTP Server, the plug-in builds the private header from the information contained in the associated request record.
- In the case of a Sun One Web server, the plug-in builds the private header from the information contained in the **auth_user** property associated with the request. The private header is usually set to the name of the local HTTP user of the Web browser, if HTTP access authorization is activated for the URL.
- In the case of a Domino Web server, the plug-in builds the private header from the information contained in the **REMOTE_USER** environment variable. The plug-in sets this variable to **anonymous** for users who have not logged in and to the *username* for users who are logged into the application.
- In the case of an Internet Information Services (IIS) Web server, the plug-in builds the private header from the information contained in the **REMOTE_USER** environment variable. The plug-in sets this variable to the name of the user as it is derived from the authorization header sent by the client.

If the private header is not being set in the Sun One, IIS, or Domino Web server plug-in, make sure the request record includes information about the user requesting the data.

If an application's call to getRemoteUser() returns a null value, or if the correct remote user information is not being added to the Web server plug-in's data structure, make sure the remote user parameter within the WebAgent is still set to **YES**. (Sometimes this parameter gets set to **NO** when service is applied.)

Web server plug-ins

Web server plug-ins enable the Web server to communicate requests for dynamic content, such as servlets, to the application server. A Web server plug-in is associated with each Web server definition. The configuration file (plugin-cfg.xml) that is generated for each plug-in is based on the applications that are routed through the associated Web server.

A Web server plug-in is used to forward HTTP requests from a supported Web server to an application server. Using a Web server plug-in to provide communication between a Web server and an application server has the following advantages:

- XML-based configuration file
- Standard protocol recognized by firewall products
- Security using HTTPS, replacing proprietary Open Servlet Engine (OSE) over Secure Sockets Layer (SSL)

Each of the supported Web server plug-ins runs on a number of operating systems. See the Supported Hardware and Software Web site for the product for the most current information about supported Web servers. This site is located at http://www.ibm.com/support/docview.wss?rs=180&uid=swg27006921.

Checking your IBM HTTP Server version

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

1. Change the directory to the installation root of the Web server.
2. Find the subdirectory that contains the executable. The executable is:
   - APACHEDFT
3. Issue the command.

   Run the STRTCPSVR command with the -V option.

   ```
   STRTCPSVR SERVER(*HTTP) HTTPSVR(APACHEDFT '-V')
   ```

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

```
Server version: Apache/2.0.43 Server built:   Nov 26 2005 15:57:01
```

In this example an IBM HTTP Server powered by Apache 2.0.43 is installed.

Creating or updating a global Web server plug-in configuration file

If all of the application servers in a cell use the same Web server to route requests for dynamic content, such as servlets, from Web applications to application servers, you can create a global Web server plug-in configuration file for that cell. The resulting plugin-cfg.xml file is located in the %was_profile_home%/config/cells directory.

You must update the global Web server plug-in configuration file whenever you:

- Change the configuration settings for an application server, cluster, virtual host or Web container transport that is part of that cell.
- Add a new application server, cluster, virtual host or Web container transport to that cell.

To update the configuration settings for a global Web server plug-in, you can either use the Update global Web server plug-in configuration page in the administrative console, or issue the following command:

```
%was_profile_home%/config/cells/GenPluginCfg.sh|bat
```

Both methods for regenerating the global Web server plug-in configuration create a plugin-cfg.xml file in ASCII format.

To use the Update global Web server plug-in configuration page in the administrative console:

1. Click **Environment > Update global Web server plug-in configuration**.
2. Click **OK** to update the plugin-cfg.xml file.
3.
4. Click **View or download the current Web server plug-in configuration file** if you want to view or download the current version of this file. You can select this option if you want to:
   - View the current version of the file before you update it.
   - View the file after it is updated.
   - Download a copy of this file to a remote machine.

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

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

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

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

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

You might need to stop the application servers in the cell and then start the application servers again before the changes to the plug-in configuration go into effect.

If the Web server is running on a remote machine, click **View or download the current Web server plug-in configuration file** to download a copy of the plugin-cfg.xml file to a that machine.

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 application server directory structures, and not assume those of the deployment manager after the plug-in is regenerated and a full synchronization occurs. The plugin-cfg.xml file is located in the application server /config/cells directory.

- **Command line**:

  At a command prompt, enter the following command to change the DeploymentManager/bin directory and type on the machine where the deployment manager is installed. This command creates or updates the plugin-cfg.xml file, and changes all of the directories in the plugin-cfg.xml file to *app_server_root* directories.

  ```
  GenPluginCfg -destination.root <app_server_root>
  ```

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

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

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

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

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

The deployment manager plugin-cfg.xml file can point to the application server directories without any conflict.

Update the global Web server plug-in configuration setting

Use this page to create or update a global plug-in configuration file. The configuration settings this file contains are based on the topology of the cell that contains the applications servers that use this Web server plug-in. The Web server plug-in configuration file settings determine whether an application server or the Web server handles user requests.

A global Web server plug-in configuration file must be regenerated whenever:

- You change the configuration settings for an application server, cluster, Web container transport, or virtual host alias that is contained in the cell.
- You add a new application server, cluster, Web container transport, or virtual host alias to the cell.

The generated `plugin-cfg.xml` file is placed in the `%was_profile_home%/config/cells` directory. If your Web server is located on a remote machine, you must manually move this file to that machine.

To view this administrative console page, click **Environment > Update global Web server plug-in configuration**

Click **OK** to update the global `plugin-cfg.xml` file.

Click **View or download the current Web server plug-in configuration file** if you want to:

- View the current version of the file before you update it.
- View the file after it is updated.
- Download a copy of this file to a remote machine.

Gskit install images files

The Global Security Kit (GSKit) installation image files for the WebSphere Web server plug-ins are packaged on the CD with the Web server plug-in files.

You can download the appropriate GSKIT file to the workstation on which your Web server is running. Use the following table to assist you in selecting the correct GSKIT installation image file.

Operating system	GSKit 7 Installation image file
Windows	No image name
AIX	gskta.rte
HP-UX	gsk7bas
Solaris Operating Environment	gsk7bas
Linux	gsk7bas_7.0.3.1.i386.rpm
Linux390	gsk7bas-7.0.3.1.s390.rpm
LinuxPPC	gsk7bas-7.0.3.1.ppc.rpm

Plug-ins: Resources for learning

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

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

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

Programming model and decisions
- Best Practice: WebSphere Plug-in Configuration Regeneration at http://www-128.ibm.com/developerworks/websphere/library/bestpractices/plug_in_configuration_regeneration.html

Programming instructions and examples
- IBM HTTP Server documentation at http://www-3.ibm.com/software/webservers/httpservers/library.html
- WebSphere Application Server education at http://www-128.ibm.com/developerworks/search/searchResults.jsp?searchType=1&searchSite=dW&searchScope=dW&query=WebSphere+education
- Listing of all IBM WebSphere Application Server Redbooks at http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere

Web server plug-in tuning tips

Balancing workloads

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

You can limit the number of connections that can be handled by an applications server. To do this:

1. Go to the **Servers** > **Application Servers** > *server_name*.
2. Under Additional Properties, click > **Web Server Plug-in properties** .
3. Select **Set limit** for the Minimum number of connections that can be handled by the Application Server field.
4. Specify in the Connections field the maximum number of connections you want to allow.
5. Then click **Apply** and **Save**.

When this maximum number of connections is reached, the plug-in, when establishing connections, automatically skips that application server, and tries the next available application server. If no application servers are available, an HTTP 503 response code will be returned to the client. This code indicates that the server is currently unable to handle the request because it is experiencing a temporary overloading or because maintenance is being performed.

The capacity of the application servers in the network determines the value you specify for the maximum number of connections. The ideal scenario is for all of the application servers in the network to be optimally utilized. For example, if you have the following environment:

- There are 10 application servers in a cluster.
- All of these application servers host the same applications (that is, Application_1 and Application_2).
- This cluster of application servers is fronted by five IBM HTTP Servers.

- The IBM HTTP Servers get requests through a load balancer.
- Application_1 takes approximately 60 seconds to respond to a request
- Application_2 takes approximately 1 second to respond to a request.

Depending on the request arrival pattern, all requests to Application_1 might be forwarded to two of the application servers, say Appsvr_1 and Appsvr_2. If the arrival rate is faster than the processing rate, the number of pending requests to Appsvr_1 and Appsvr_2 can grow.

Eventually, Appsvr_1 and Appsvr_2 are busy and are not able to respond to future requests. It usually takes a long time to recover from this overloaded situation.

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

Limiting the number of connections that can be established with an application server works best for Web servers that follow the threading model instead of the process model, and only one process is started.

The IBM HTTP Server V6.1 follows the threading model. To prevent the IBM HTTP Server from starting more than one process, change the following properties in the Web server configuration file (httpd.conf) to the indicated values:

```
ServerLimit           1
ThreadLimit           4000
StartServers          1
MaxClients            1024
MinSpareThreads       1
MaxSpareThreads       1024
ThreadsPerChild       1024
MaxRequestsPerChild   0
```

Private headers

A Web server plug-in can use private headers to forward requests for dynamic content, such as servlets, to the application server.

After you configure a Web server plug-in, in addition to regular plug-in functions, you can use private headers as a mechanism for forwarding proxy information from the plug-in to an application server. This information is not normally included in HTTP requests.

Private headers are implemented as a set of HTTP request header name and value pairs that the plug-in adds to the HTTP request header before the request is forwarded to an application server. The application server's Web container removes this information from the header and then processes this information.

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

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

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

plugin-cfg.xml file

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

Starting with Version V6.0.1, the plug-in configuration file is generated in ASCII format (ISO-98859-1). (Previously the configuration file was generated in EBCDIC format.) If you need to edit this file, issue the following command to convert the file to EBCDIC format:

```
> iconv  -f ISO8859-1 -t IBM-1047 plugin-cfg.xml.ASCII > plugin-cfg.xml.EBCDIC
```

Edit the file, and then issue the following command to convert it back to ASCII format:

```
> iconv  -f IBM-1047 -t ISO8859-1  plugin-cfg.xml.EBCDIC > plugin-cfg.xml.ASCII
```

CAUTION:
Use the administrative console to set these properties for a given Web server definition. Any manual changes you make to the plug-in configuration file for a given Web server are overridden whenever the file is regenerated.

Config (required)

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

IgnoreDNSFailures

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

RefreshInterval

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

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

ASDisableNagle

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

The value can be true or false.

IISDisableNagle

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

The value can be true or false.

AppServerPortPreference

This attribute is used to specify which port number the Application Server should use to build URI's for a sendRedirect. The following values can be specified:

- webserverPort if the port number from the host header of the HTTP request coming in is to be used.
- hostHeader if the port number on which the Web server received the request is to be used.

The default is hostHeader.

ResponseChunkSize

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

The ResponseChunkSize attribute lets you specify the maximum chunk size to use when reading the response body. For example, Config ResponseChunkSize=″N″>, where N equals the chunk size in kilobytes.

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

The default chunk size is 64k.

AcceptAllContent

Specifies whether or not users can include content in POST, PUT, GET, and HEAD requests when a Content-Length or Transfer-encoding header is contained in the request header. You can specify one of the following values for this attribute:

- True if content is to be expected and read for all requests
- False if content only is only to be expected and read for POST and PUT requests.

False is the default.

ChunkedResponse

Specifies whether the plug-in should chunk the response to the client when a Transfer-Encoding : Chunked response header is present in the response.

This attribute only applies to the IIS, IPlanet, and Domino Web servers. The IBM HTTP Server automatically handles the chunking of the response to the client.

You can specify one of the following values for this attribute:

- true if the plug-in is to chunk the response to the client when a Transfer-Encoding : Chunked response header is present in the response.
- false if the response is not to be chunked.

false is the default.

IISPluginPriority

Specifies the priority in which the IIS Web server loads the WebSphere Web server plug-in. You can specify one of the following values for this attribute:

- High
- Medium
- Low

The default value is High.

NOTES:

- The IIS Web server uses this value during startup. Therefore, the Web server must be restarted before this change will take effect.
- The default value of High ensures that all requests are handled by the WebSphere Web server plug-in before they are handled by any other filter/extensions. If problems occur while using a priority of Medium or Low, you will have to rearrange the order or change the priority of the interfering filter/extension.

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

For example, you might specify the following:

```
<Log LogLevel="Error" Name="/opt/WebSphere/AppServer60/logs/http_plugin.log"/>
```

Name (exactly one attribute for each Log)

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

If the file does not exist then it will be created. If the file already exists then it will be opened in append mode and the previous plug-in log messages will remain.

LogLevel (zero or one attribute for each Log)

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

- Trace. All of the steps in the request process are logged in detail.
- Stats. The server selected for each request and other load balancing information relating to request handling is logged.
- Warn. All warning and error messages resulting from abnormal request processing are logged.
- Error. Only error messages resulting from abnormal request processing are logged.
- Debug. All of the critical steps performed in processing requests are logged.
- Detail. All of the information about requests and responses are logged.

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

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

Property Name="esiEnable" Value="*true/false***"**

Used to enable or disable the Edge Side Include (ESI) processor. If the ESI processor is disabled, the other ESI elements in this file are ignored.

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

Property Name="esiMaxCacheSize" Value="*interger***"**

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

Property Name="ESIInvalidationMonitor" Value="*true/false***"**

Used to indicate whether or not the ESI processor should receive invalidations from the Application Server.

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

Property Name="FIPSEnable" Value="*true/false***"**

Used to indicate whether or not the Federal Information Processing Standard (FIPS) is enabled for making secure (SSL) connections to the Application Server. This property should be set to true, if FIPS is enabled on the Application Server..

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

ServerCluster (one or more elements for each Config)

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

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

Following is an example of a ServerCluster element

```
<ServerCluster Name="Servers">
<ClusterAddress Name="ClusterAddr">
<Transport Hostname="192.168.1.2" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.2" Port="9443" Protocol="HTTPS">
<Property Name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
```

```
<Property Name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Transport>
</ClusterAddress>
<Server Name="Server1">
<Transport Hostname="192.168.1.3" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.3" Port="9443" Protocol="HTTPS">
<Property Name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property Name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Transport>
</Server>
<Server Name="Server2">
<Transport Hostname="192.168.1.4" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.4" Port="9443" Protocol="HTTPS">
<Property Name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property Name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Transport>
</Server>
<Server Name="Server3">
<Transport Hostname="192.168.1.5" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.5" Port="9443" Protocol="HTTPS">
<Property Name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property Name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Transport>
</Server>
<PrimaryServers>
<Server Name="Server1"/>
<Server Name="Server2"/>
</PrimaryServers>
<BackupServers>
<Server Name="Server3"/>
</BackupServers>
</ServerCluster>
```

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

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

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

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

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

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

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

The value can be true or false.

PostSizeLimit (zero or one attribute for each ServerCluster)
The maximum number of bytes of request content allowed in order for the plug-in to

attempt to send the request to an application server. If a request is received that is greater than this size, the plug-in fails the request. The default value is -1 bytes, which indicates that there is no limit for the post size.

Server (one or more elements for each ServerCluster)

A WebSphere Application Server instance that is configured to handle requests routed to it given the routing rules of the plug-in configuration. The Server should correspond to an application server running on either the local machine or a remote machine.

Name (exactly one attribute for each Server)

The administrative or logical name for the server.

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)

Specifies the weight associated with this server when the plug-in does weighted Round Robin load balancing. The starting value for a server can be any integer between 0 and 20. However, zero should be specified only for a server that is shut down.

The algorithm for this attribute decrements all weights within the server cluster until all weights reach zero. After the weight specified for a particular server reaches zero, no more requests are routed to that server until all servers in the cluster have a weight of zero. After all servers reach zero, the weights for all servers in the cluster are reset and the algorithm starts over.

When a server is shut down, it is recommended that you set the weight for that server to zero. The plug-in can then reset the weights of the servers that are still running, and maintain proper load balancing.

ConnectTimeout (zero or one attribute for each Server)

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

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

ExtendedHandshake (zero or one attribute for each Server)

The ExtendedHandshake attribute is used when a proxy firewall is between the plug-in and the application server. In such a case, the plug-in is not failing over, as expected.

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

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

The value can be true or false.

MaxConnections (one element for each Server)

The MaxConnections attribute is used to specify the maximum number of pending connections to an Application Server that can be flowing through a Web server process at any point in time.

For example, assuming that:

- The application server is fronted by 5 nodes that are running an IBM HTTP Server.
- Each node starts 2 processes.
- The MaxConnections attribute is set to 50.

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

This attribute is not necessary on the z/OS platform. The z/OS controller working in conjunction with WLM, handles new connections dynamically.

By default, MaxConnections is set to -1. If this attribute is set to either zero or -1, there is no limit to the number of pending connections to the Application Servers.

Transport (one or more elements for each Server)

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

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

Hostname (exactly one attribute for each Transport)

The host name or IP address of the machine on which the WebSphere application server instance is running.

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

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

Property (zero, one, or more elements for each Transport)
When the Protocol of the Transport is set to HTTPS, use this element to supply the various initialization parameters, such as password, keyring and stashfile. for example, the portion of the plugin_cfg.xml file containing these elements might look like the following:

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

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

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

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

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

ServerIOTimeout
The ServerIOTimeout attribute of a server element enables the plug-in to set a time out value, in seconds, for sending requests to and reading responses from the application server. If a value is not set for the ServerIOTimeout attribute, the plug-in, by default, uses blocked I/O to write request to and read response from the application server until the TCP connection times out. For example, if you specify:

```
<Server Name="server1" ServerIOTimeout=300>
```

In this case, if an application server stops responding to requests, the plug-in waits 300 seconds (5 minutes) before timing out the TCP connection. Setting the ServerIOTimeout attribute to a reasonable value enables the plug-in to time out the connection sooner, and transfer requests to another application server when possible.

When selecting a value for this attribute, remember that sometimes it might take a couple of minutes for an application server to process a request. Setting the value of the ServerIOTimeout attribute too low could cause the plug-in to send a false server error response to the client.

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

Important: If you include a ClusterAddress tag, you must include the Name attribute on that tag. The plug-in uses the name attribute to associate the cluster address

with the correct host and port. If you do not specify the Name attribute, the plug-in assigns the cluster address the name that is specified for the server that is using the same host and port.

```
<ClusterAddress Name="MyClusterAddr">
<Transport Hostname="192.168.1.2" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.2" Port="9443" Protocol="HTTPS">
</ClusterAddress>
```

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

PrimaryServers (zero or one element for each server cluster)

Specifies a list of servers to which the plug-in routes requests for this cluster. If a list of primary servers is not specified, the plug-in routes requests to servers defined for the server cluster.

BackupServers (zero or one element for each server cluster)

Specifies a list of servers to which requests should be sent to if all servers specified in the primary servers list are unavailable. The plug-in does not load balance across the backup servers, but traverses the list in order until no servers are left in the list or until a request is successfully sent and a response received from an application server.

VirtualHostGroup

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

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

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

Name (exactly one attribute for each VirtualHostGroup)

The logical or administrative name to be used for this group of virtual hosts.

VirtualHost (one or more elements for each VirtualHostGroup)

The name used for a virtual or real machine used to determine if incoming requests should be handled by WebSphere Application Server or not. Use this element to specify host names that will be in the HTTP Host header which should be seen for requests that need to be handled by the application server. You can specify specific host names and ports that incoming requests will have or specify an asterisk (*) for either the host name, port, or both.

Name (exactly one attribute for each VirtualHost)

The actual name that should be specified in the HTTP Host header in order to match successfully with this VirtualHost.

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

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

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

UriGroup

A group of URIs that will be specified on the HTTP request line. The same application server must

be able to handle the URIs. The route will compare the incoming URI with the URIs in the group to determine if the application server will handle the request.

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

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

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

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

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

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

See the description of the CloneID attribute for additional session affinity information.

AffinityURLIdentifier (zero or one attribute for each Uri)
The name of the identifier the plug-in should use when trying to determine if the inbound request has affinity specified in the URL to a particular clone. The default value is **jsessionid**.

See the description of the CloneID attribute for additional session affinity information.

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

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

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

Following is an example of this element:

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

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

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

UriGroup (zero or one attribute for each Route)

The group of URIs to use for determining the route. The incoming URI for the request is matched to the defined URIs in this group to determine if this request should be handled by the application server.

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

ServerCluster (exactly one attribute for each Route)

The cluster to which to send request that successfully match the route.

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

RequestMetrics

This element is used to determine if request metrics is enabled, and how to filter the requests based on the Internet protocol (IP) and Uniform Resource Identifiers (URI) filters when request metrics is enabled.

Following is an example of this element:

```
<RequestMetrics armEnabled="false"  loggingEnabled="true"
   rmEnabled="false" traceLevel="PERF_DEBUG">
```

armEnabled (zero or one attribute for RequestMetrics)

This attribute indicates whether the ARM 4 agent is enabled in the plug-in. When it is set to true, the ARM 4 agent will be called.

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

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

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

loggingEnabled (exactly one attribute for RequestMetrics)

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

rmEnabled (exactly one attribute for RequestMetrics)

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

traceLevel (exactly one attribute for RequestMetrics)

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

filters (zero, one, or two attributes for RequestMetrics)

When rmEnabled is true, the filters control which requests are traced.

enable (exactly one attribute for each filter)

When enable is true, the type of filter is on and requests must pass the filter.

type (exactly one attribute for each filter)

There are two types of filters: SOURCE_IP (for example, client IP address) and URI. For the SOURCE_IP filter type, requests are filtered based on a known IP address. You can specify a mask for an IP address using the asterisk (*). If the asterisk is used, the asterisk must always be the last character of the mask, for example 127.0.0.*, 127.0.*, 127*. For

performance reasons, the pattern matches character by character, until either an asterisk is found in the filter, a mismatch occurs, or the filters are found as an exact match.

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

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

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

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

Setting up a local Web server

This topic describes how to install the Web server and the Web server plug-in on the machine where you installed WebSphere Application Server.

Important: Non-IBM HTTP Server Web servers must reside on a managed node to facilitate plug-in administration functions and generation and propagation of the `plugin-cfg.xml` file.

You can define a locally-installed Web server on an unmanaged or managed node. If the Web server is defined on an unmanaged node, the administrative functions are handled through the IBM HTTP Server administration server. If the Web server is defined on a managed node, the administrative functions of the Web server are handled through the WebSphere Application Server node agent, which is beneficial.

The following steps create a Web server definition in the default profile.

1. Install your WebSphere Application Server product.
2. Install IBM HTTP Server or another supported Web server.
3. Install the binary plug-in module using the Plug-ins installation wizard.
4. Complete the setup by creating the Web server definition using the WebSphere Application Server administrative console, or run the plug-in configuration script. The creation of this object is exclusive of the Web server installation.

   Select one of the following options:

   - **Using the administrative console.** You must create a Web server definition on an existing application server or unmanaged node.
     a. Click **Servers > Web servers > New** and use the **Create new Web server entry** wizard to create the Web server definition.
     b. Select the appropriate node.
     c. Enter the Web server properties:
        – Type: The Web server vendor type
        – Port: The existing Web server port (default: 80)
        – Installation path: The Web server installation path. This field is required for IBM HTTP Server only.
        – Service name (Windows operating systems): The Windows operating system service name of the Web server. The default is `IBMHTTPServer6.0`.
        – Use secure protocol: Use the HTTPS protocol to communicate with the Web server. The default is `HTTP`.
        – Plug-in installation location: The directory path in which the plug-in is installed.
     d. Select a template. Select a system template or a user-defined template for the Web server you want to create.

e. Confirmation of Web server creation.

- **Running the plug-in configuration script.**

  If you install the plug-in, save the plug-in configuration script to run after you create a managed node, otherwise an error occurs. Wait until the script runs successfully and creates the Web server definition on the managed node and node synchronization occurs before starting the Web server.

  Adding the node starts the node agent process. If the node agent is not running, start the node. **Tip**: If you want the Web server to handle requests for an application for multiple managed nodes, install the application on each managed node and on the Web server definition. The script already contains all of the information that you must gather when using the administrative console option.

You can configure non-IBM HTTP Server Web servers as a remote Web server on unmanaged nodes, or as a local Web server on managed nodes. For a non-IBM HTTP Server Web server on a managed node, the following functions are supported:

- Generation of the plug-in configuration, based on WebSphere Application Server repository changes.
- Propagation of the `plugin-cfg.xml` file, based on using node synchronization with the WebSphere Application Server node. Node synchronization is necessary in order to propagate configuration changes to the affected node or nodes.

  The `plugin-cfg.xml` file is propagated to the application server node repository tree from the deployment manager repository.

  Important: The `plugin-cfg.xml` file is propagated to the application server node repository tree. This is not the default `plugin-cfg.xml` file installation location. Changes may have to be made to non-IBM HTTP Server Web server configuration files to update the location of the `plugin-cfg.xml` file that is read by the plug-in module.

  For example, Internet Information Services (IIS) has a file name called `plugin-cfg.loc`, which is read by the IIS plug-in modules to determine the location of the `plugin-cfg.xml` file. The `plugin-cfg.loc` file has to be updated to reflect the `plugin-cfg.xml` file location in the application server node repository.

  Other non-IBM HTTP Server Web servers have different methods to specify the location of the `plugin-cfg.xml` file for the plug-in module. However, in order for propagation to work, update the location to reflect the location in the application server node repository.

The following functions are not supported on a managed node:

- Starting and stopping the Web server.
- Viewing and editing the configuration file.
- Viewing the Web server logs.

For a non-IBM HTTP Server Web Server on an unmanaged node, you can generate plug-in configuration, based on WebSphere Application server repository changes. The following functions are not supported on an unmanaged node for a non-IBM HTTP Server Web server:

- Starting and stopping the Web server.
- Viewing and editing the configuration file.
- Viewing the Web server logs.
- Propagation of the Web server `plugin-cfg.xml` file.

Setting up a remote Web server

This topic describes how to create a Web server definition in the administrative console when the Web server and the Web server plug-in for WebSphere Application Server are on one machine and the application server is on another.

Important: Non-IBM HTTP Server Web servers must reside on a managed node to facilitate plug-in administration functions and generation and propagation of the `plugin-cfg.xml` file.

You can choose a remote Web server installation if you want the Web server on the outside of a firewall and WebSphere Application Server on the inside a firewall. You must create a remote Web server on an unmanaged node. Unmanaged nodes are nodes without node agents. Since there is no WebSphere Application Server, or node agent on the machine that the node represents, there is no way to administer a Web server on that unmanaged node unless the Web server is IBM HTTP Server. In this case, there is an administration server that will facilitate administrative requests such as start and stop, view logs, and view and edit the `httpd.conf` file. The following steps will create a Web server definition in the default profile.

1. Install your WebSphere Application Server product.
2. Install IBM HTTP Server or another supported Web server.
3. Install the binary plug-in module using the Plug-ins installation wizard.
4. Complete the setup by creating the Web server definition. You can use the WebSphere Application Server administrative console or run the Plug-in configuration script:
   - **Using the administrative console:**
     a. Click **System Administration > Nodes > Add Node** to create an unmanaged node in which to define a Web server in the topology.
     b. Click **Servers > Web servers > New** to launch the **Create new Web server entry** wizard. You will create the new Web server definition using this wizard. The wizard values are as follows:
        1) Select appropriate node
        2) Enter Web server properties:
           – **Type**: The Web server vendor type.
           – **Port**: The existing Web server port. The default is 80.
           – **Installation Path**: The Web server installation path. This field is required field for IBM HTTP Server only.
           – **WINDOWS Service Name**: The windows operating system service name of the Web server. The default is `IBMHTTPServer6.0`.
           – **Use secure protocol**: Use the HTTPS protocol to communicate with the Web server. The default is `HTTP`.
           – **Plug-in installation location**: The directory path where the plug-in is installed.
        3) Enter the remote Web server properties. The properties for the IBM HTTP Server administration server follow:
           – **Port**: The administration server port. The default is `8008`.
           – **User ID**: The user ID that is created using the htpasswd script.
           – **Password**: The password that corresponds to the user ID created with the `htpasswd` script.
           – **Use secure protocol**: Use the HTTPS protocol to communicate with the administration server. The default is `HTTP`.
        4) Select a Web server template. Select a system template or a user-defined template for the Web server you want to create.
        5) Confirmation of Web server creation.
   - **Running the Plug-in configuration script.**
5. Run the `setupadm` script on Linux and UNIX platforms. The administration server requires read and write access to configuration files and authentication files to perform Web server configuration data administration. You can find the `setupadm` script in the *<IHS_install_root>*`/bin` directory. The administration server has to execute `adminctl restart` as root to perform successful restarts of IBM HTTP Server. In addition to the Web server files, you must manually change the permissions to the targeted plug-in configuration files.

   The `setupadm` script prompts you for the following input:

- User ID - The user ID that you use to log on to the administration server. The script creates this user ID.
- Group name - The administration server accesses the configuration files and authentication files through group file permissions. The script creates the specified group through this script.
- Directory - The directory where you can find configuration files and authentication files.
- File name - The following file groups and file permissions change:
  - Single file name
  - File name with wildcard
  - All (default) - All of the files in the specific directory
- Processing - The `setupadm` script changes the group and file permissions of the configuration files and authentication files.

In addition to the Web server files, you must change the permissions to the targeted plug-in configuration files. See Setting permissions manually for instructions.

6. Run the `htpasswd` script on Linux, UNIX and Windows platforms. The administration server is installed with authentication enabled and a blank `admin.passwd` password file . The administration server will not accept a connection without a valid user ID and password. This is done to protect the IBM HTTP Server configuration file from unauthorized access.

   Launch the **htpasswd** utility that is shipped with the administration server. This utility creates and updates the files used to store user names and password for basic authentication. Locate **htpasswd** in the `bin` directory.

   - On Windows operating systems: `htpasswd -cm <install_dir>\conf\admin.passwd [login name]`
   - On Linux and UNIX platforms: `./htpasswd -cm <install_dir>/conf/admin.passwd [login name]`

   where *<install_dir>* is the IBM HTTP Server installation directory and *[login name]* is the user ID that you use to log into the administration server. The [login name] is the user ID that you entered in the user ID field for the remote Web server properties in the administrative console.
7. Start IBM HTTP Server. Refer to Starting the IBM HTTP administration server on Windows operating systems or Starting the IBM HTTP administration server on Linux and UNIX platforms for instructions.

You can configure non-IBM HTTP Server Web servers as a remote Web server on unmanaged nodes, or as a local Web server on managed nodes. For a non-IBM HTTP Server Web server on a managed node, the following functions are supported:

- Generation of the plug-in configuration, based on WebSphere Application Server repository changes.
- Propagation of the `plugin-cfg.xml` file, based on using node synchronization with the WebSphere Application Server node. Node synchronization is necessary in order to propagate configuration changes to the affected node or nodes.

  The `plugin-cfg.xml` file is propagated to the application server node repository tree from the deployment manager repository.

  Important: The `plugin-cfg.xml` file is propagated to the application server node repository tree. This is not the default `plugin-cfg.xml` file installaion location. Changes may have to be made to non-IBM HTTP Server Web server configuration files to update the location of the `plugin-cfg.xml` file that is read by the plug-in module.

  For example, Internet Information Services (IIS) has a file name called `plugin-cfg.loc`, which is read by the IIS plug-in modules to determine the location of the `plugin-cfg.xml` file. The `plugin-cfg.loc` file has to be updated to reflect the `plugin-cfg.xml` file location in the application server node repository.

  Other non-IBM HTTP Server Web servers have different methods to specify the location of the `plugin-cfg.xml` file for the plug-in module. However, in order for propagation to work, update the location to reflect the location in the application server node repository.

The following functions are not supported on a managed node:

- Starting and stopping the Web server.

- Viewing and editing the configuration file.
- Viewing the Web server logs.

For a non-IBM HTTP Server Web Server on an unmanaged node, you can generate plug-in configuration, based on WebSphere Application server repository changes. The following functions are not supported on an unmanaged node for a non-IBM HTTP Server Web server:

- Starting and stopping the Web server.
- Viewing and editing the configuration file.
- Viewing the Web server logs.
- Propagation of the Web server `plugin-cfg.xml` file.

Web server definition

To administer or manage a Web server using the administrative console, you must create a Web server definition or object in the WebSphere Application Server repository.

The creation of this object is exclusive of the actual installation of a Web server. The Web server object in the WebSphere Application Server repository represents the Web server for administering and managing the Web server from the administrative console. The Web server object contains Web server properties, for example, installation root, port, configuration file paths, and log file paths. In addition to Web server properties, the Web server contains a plug-in object. The plug-in object contains properties that define the `plugin-cfg.xml` file.

The definitions of the Web server object are made using the **wsadmin** command or the administrative console. You can also define a Web server object in the WebSphere Application Server repository using the profile create script during installation, a `.jacl` script, and by using the administrative console wizard.

You have three types of WebSphere Application Server nodes upon which you can create a Web server. The type depends on the version of WebSphere Application Server, as follows:

- **Managed node**. A node that contains a node agent. This node can exist only in a deployment manager environment. The importance of defining a Web server on a managed node is that the administration and configuration of the Web server is handled through the node agent from the administrative console. Support for administration and configuration through the administrative console is limited to IBM HTTP Server only. Non-IBM HTTP Server Web servers must be on a managed node to handle plug-in administrative functions and the generation and propagation of the `plugin-cfg.xml` file.
- **Stand-alone node**. A node that does not contain a node agent. This node usually exists in an Express or Base environment. A stand-alone node can become a managed node in a deployment manager environment after the node is federated . A stand-alone node does not contain a node agent, so to administer and manage IBM HTTP Server, there must be an IBM HTTP Server administration server installed and running on the stand-alone machine that the node represents. IBM HTTP Server ships with the IBM HTTP Server administration server and is installed by default. Support for administration and configuration through the administrative console is limited to IBM HTTP Server only.
- **Unmanaged node**. A node that is not associated with a WebSphere Application Server node agent. This node cannot be federated. Typically, the unmanaged or dummy node represents a remote machine that does not have WebSphere Application Server installed. However, you can define an unmanaged node on a machine where WebSphere Application Server is installed and where a node agent is present. This node can exist in an Express, Base, or deployment manager environment. A dummy node does not contain a node agent, so to administer and manage IBM HTTP Server, an IBM HTTP Server administration server must be installed and running on the stand-alone machine that the node represents. Support for administration and configuration through the administrative console is limited to IBM HTTP Server only.

Administration of the Web server

The IBM HTTP Server Web server is administered and managed on the Web server collection panel in the WebSphere Application Server administrative console. You can start and stop IBM HTTP Server from the administrative console.

In WebSphere Application Server V6.1, you can create a Web server on a stand-alone node, enabling the Web server definition created on the stand-alone node to be included in the Deployment Manager's managed node when it is federated. Web servers that are defined on a stand-alone node are managed just as a Web server that is defined on an unmanaged node. An IBM HTTP Server Web server that is defined on a stand-alone node is managed by the IBM HTTP Server administration server. Non-IBM HTTP Server Web servers are not managed because no administrative agent exists to handle administration management.

Editing the Web server type

This topic provides information on how to change the type of Web server.

If you install a Web server that is different from the one that is currently installed, you can modify the Web server type from IBM HTTP Server to a non-IBM HTTP Server and vice versa, rather than delete the Web server and create a new Web server definition. If you change the Web server type from IBM HTTP Server to non-IBM HTTP Server Web server, the administration capabilities are lost accordingly.

1. From the WebSphere Application Server administrative console, click **Servers > Web servers**.
2. Select the server that you want to modify.
3. On the Web server configuration panel, change your Web server by selecting an option from the Type drop-down menu. If you are changing from a non-IBM HTTP Server to an IBM HTTP Server, you are also prompted for information such as IBM HTTP Server administration server port, user ID, and IBM HTTP Server administration server password.
4. Click **Apply**.

You can verify your changes on the Web servers collection panel. The Web server type displays in the Web Server Type column.

Web server collection

Use this page to view configure, manage, and view information about your Web servers.

Web servers

To view this administrative console page click **Servers > Web servers**.

To create a new Web server, click the **New** button to launch the **Create new Web server entry** wizard. To manage an installed Web server, select the check box beside the application name in the list and click a button:

Button	Resulting Action
Generate Plug-in	When the plug-in configuration service is enabled, a plug-in configuration file is automatically generated for a Web server whenever: • The WebSphere Application Server administrator defines new Web server. • An application is deployed to an Application Server. • An application is uninstalled. • A virtual host definition is updated and saved.

Button	Resulting Action
Propagate Plug-in	Choosing this action will copy the `plugin-cfg.xml` file from the local directory where the Application Server is installed to the remote machine. If you are using IBM HTTP Server V6 for your Web server, WebSphere Application Server can automatically propagate the plug-in configuration file to remote machines provided there is a working HTTP transport mechanism to propagate the file.
New	Launches the wizard to create a new Web server entry.
Delete	Deletes one or more of the selected Web server entries.
Templates...	Opens the Web server templates list panel. From this panel you can create a new template or delete existing templates.
Start	Starts one or more of the selected Web servers.
Stop	Stops one or more of the selected Web servers.
Terminate	Terminates one or more of the selected Web servers.

Name
Specifies a logical name for the Web server. This can be the host name of the machine, or any name you choose.

Web server type
Indicates the type of Web Server you are using.

Node
Specifies the name of the node on which the Web server is defined.

Version
Specifies the version of the WebSphere Application Server node on which the Web server is defined.

Status
Indicates whether the Web server is started, stopped, or unavailable.

If IBM HTTP Server is defined on an Unmanaged node, you will need to start the IBM HTTP Server administration server before you can start and stop IBM HTTP Server.

Note that if the status is *Unavailable*, the node agent or IBM HTTP Server administration server is not running in that node, and you must start the node agent before you can start the Web server.

Web server configuration

Use this page to configure Web server properties.

Web servers

To view this administrative console page click **Servers > Web servers >** *Web_server_name*.

Web server name
Specifies a logical name for the Web server.

Type
Specifies the vendor of the Web server. The default value is IBM HTTP Server.

The options for the type of Web servers are:

- IHS
- APACHE
- IIS
- SUNJAVASYSTEM
- DOMINO

Port
The port from which to ping the status of the Web server. This field is required.

You can use the WebSphere Application Server administrative console to check if the Web server is started by sending a ping to attempt to connect to the Web server port that is defined. In most cases the port is 80. If you have a firewall between the Web servers and application servers, you will not use port 80 on the firewall between the two systems. In most cases, your port will be different, such as 9080 or 9443. You should set the alternate ports for the Web server using the WebSphere Application Server administrative console, then set that port to the Web server to listen on, in addition to the typical port 80 and 443.

Installation path
Enter the fully qualified path where the Web server is installed. This field is required if you are using IBM HTTP Server. For all other Web Servers, this field is not required. If you enable any administrative function for non-IBM HTTP Server Web servers, the installation path will be necessary.

Configuration file name
There are two ways to view or modify the contents of the configuration file:

1. Click **Edit** to view the configuration file. You will be able to make modifications from this view. This is valid for IBM HTTP Server only.
2. Click **Configuration file** under Additional properties. You will be able to make modifications from this view. This is valid for IBM HTTP Server only.

Service name - Windows operating systems only
Specifies the Windows operating system name for the Web server. The name is the service name and you can find it by opening the **General** properties tab of the Web server service name.

Web server log file

Use this page to view the log file for your Web server.

To view this administrative console page, click **Servers > Web Servers >** *Web_server_name* > **Log file**.

Web server log file configuration

Access log file name
Any request that is made to the Web server displays in this file.

Error log file name
Any error that occurs in the Web server displays in this file.

Web server log file runtime

Access log file name
Click **View** to display the contents of this file.

Error log file name
Click **View** to display the contents of this file.

Web server custom properties

Use this page to view and manage arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties.

The administrative console contains several Custom Properties pages that work similarly. To view one of these administrative pages, click a Custom Properties link.

Web servers
To view this administrative console page click **Servers > Web Servers >** *Web_server_name* **> Custom properties**.

Name	Specifies the name (or key) for the property.
Value	Specifies the value paired with the specified name.
Description	Provides information about the name-value pair.

Remote Web server management
Use this page to configure the properties of an IBM HTTP Server Web server that is created on an unmanaged node.

Create a new, unmanaged node by clicking **System administration > Nodes > Add Node**. Create a Web server using the newly-created, unmanaged node by clicking **Servers > Web Servers > New**. To view the administrative console page for Remote Web server management, click **Servers > Web Servers >** *Web_server_name* **> Remote Web server management**.

Web servers

Port	Indicates the port to access the administration server (default is 8008). The HTTP administration server port is 2001 on the iSeries platform.
Use SSL	Specifies if the port is secure.
User ID	Specifies a user ID in the <*install_dir*>/conf/admin.passwd file. Create this with the htpasswd script file, located in the <*install_dir*>/bin directory.
Password	Specifies a password in the <*install_dir*>/conf/admin.passwd file. Create this with the htpasswd script file, located in the <*install_dir*>/bin directory. On the iSeries platform, there is no htpasswd script in the <*install_dir*>/bin directory. The HTTP administrator is typically created as an iSeries SECOFR profile.

Web server configuration file
Use this page to view or modify the contents of the Web server configuration file in your Web browser.

Web servers

If you have made changes to the configuration file you will need to restart your Web server, in order for the changes to take effect.

Global directives
Use this page to configure the global directives for your Web server.

To view this administrative console page, click **Servers > Web Servers >** *Web_server_name* **> Configuration settings > Global directives**.

Security enabled
Specifies if security is enabled in your Web server.

Server name
Specifies the hostname that the Web server uses to identify itself.

Listen port
Specifies the port on which your Web server will listen for requests.

Document root
The directory where the Web server will serve files.

Keystore filename
Specifies the name you have assigned to your keystore.

Keystore directory
Specifies the target directory of your keystore on the machine where the Web server is installed.

SSL Version 2 timeout
Specifies the SSL Version 2 timeout.

SSL Version 3 timeout
Specifies the SSL Version 3 timeout.

Keystore certificate label
Specifies the keystore certificate label. The certificate label specified here will be the certificate used in secure communication for this virtual host.

Virtual hosts collection
Use this page create or edit virtual hosts for your Web server.

To view this administrative console page, click **Servers > Web Servers >** *Web_server_name* > **Configuration settings > Virtual hosts**.

To create a new virtual host, click the **New** button to launch the virtual host configuration panel. To delete an existing virtual host, select the check box beside the virtual host in the list and click **Delete** .

IP address:Port
The IP address and port number of your virtual host for the specified Web server.

Server name
Specifies the name of your virtual host for the specified Web server.

Security enabled
Specifies whether or not security is enabled for your virtual host for the specified Web server. The values are **true** or **false**.

Virtual hosts detail
Use this page create or edit virtual hosts for your Web server.

To view this administrative console page, click **Servers > Web Servers >** *Web_server_name* > **Configuration settings > Virtual hosts > New**.

Security enabled
Specifies whether or not security is enabled for your virtual host for the specified Web server. Check the box to enable security

IP address
The IP address of your virtual host for the specified Web server.

Port
The port number of your virtual host for the specified Web server.

Server name
Specifies the name of your virtual host for the specified Web server.

Document root
Specifies the location of the `htdocs` directory for your Web server.

Keystore filename
Specifies the name you have assigned to your keystore.

Keystore directory
Specifies the target directory for the key store file on the machine where your Web Server is installed.

Keystore certificate label
Specifies the keystore certificate label. The certificate label specified here is the certificate that used in secure communication for this virtual host.

Chapter 3. Creating and deleting profiles

This topic describes how to create and delete profiles. A profile is the set of files that define the runtime environment. At least one profile must exist to run WebSphere Application Server.

Profiles are created by default when you install various product features. This task assumes a basic familiarity with the manageprofiles command to create additional profiles.

You usually create a profile when you install the product. Depending on which WebSphere Application Server product you have, you might create additional profiles.

You cannot use the Profile Management tool to create profiles on

- 64-bit platforms

Default profiles are created automatically when you install a product feature.

When you federate WebSphere Application Server into a Network Deployment cell, a federated default is created.

You can delete profiles through the **manageprofiles** command or by other means if necessary. You might delete a profile if the configuration that you specified in the profile is not what you want.

Perform any of the following tasks to create or delete profiles.

- Create default profiles.

  This topic describes how to create default profiles for WebSphere Application Server, Web servers, and application clients.
- Create profiles through commands.

  This topic describes how to set up and use the profile environment through commands.
- Delete a profile.

  This topic discusses how to delete a profile with and without the **manageprofiles** command.

You created a profile to manage your WebSphere Application Server runtime environment and might have started an Application Server, depending on the tasks you followed. You might have deleted a profile.

Depending on the tasks that you completed, you can start WebSphere Application Server, or proceed to other tasks, such as deploying an application.

Profile concepts

The WebSphere Application Server profile defines the runtime environment. The profile includes all of the files that the server processes in the runtime environment and can change. This topic discusses the main terms and concepts that are associated with profiles.

Core product files

The core product files are the shared product binaries, which are shared by all profiles.

The directory structure for the product has two major divisions of files in the installation root directory for the product:

- The core product files are shared product binary files that do not change unless you install a refresh pack, a fix pack, or an interim fix. Some log information is also updated.

  The default installation location for the core product files is the *app_server_root* directory.

- The *user_data_root*/profiles directory is the default directory for creating profiles.

When it is desirable to have binaries at different service levels, you must use a separate installation of WebSphere Application Server for each service level.

The configuration for every defined Application Server process is within the profiles directory unless you specify a new directory when you create a profile. These files change as often as you create a new profile, reconfigure an existing profile, or delete a profile.

If you put a profile in an installation root directory, a risk exists that the profile might be damaged or destroyed by routine system maintenance.

Why and when to create a profile

The **manageprofiles** command-line tool defines each Application Server profile for the product.

Run the command line tool each time that you want to create a stand-alone Application Server.

Administration is greatly enhanced when using profiles instead of multiple product installations. Not only is disk space saved, but updating the product is simplified when you maintain only a single set of product core files. Also, creating new profiles is faster and less prone to error than full product installations, allowing a developer to create separate profiles of the product for development and testing.

You can run the manageprofiles command to create a new Application Server environment on the same machine as an existing one. Simply define unique characteristics (such as profile name and node name) for the new profile. Each profile has its own administrative console and administrative scripting interface.

Profile types

Templates for each profile are located in the *app_server_root*/profileTemplates directory.

Within this directory are various directories that correspond to different profile types and that vary with the type of product installed. The directories are the paths that you indicate while using the manageprofiles command with the -templatePath option. You can also specify profile templates that lie outside the installation root, if you happen to have any.

See the -templatePath parameter description in the Manageprofiles command topic in the *Using the administrative clients* PDF for more information.

The manageprofiles command in the Network Deployment product can create the following types of profiles:

Deployment manager profile

The basic function of the deployment manager is to deploy applications to a cell of Application Servers, which it manages. Each Application Server that belongs to the cell is referred to as a *managed node*.

Specify *app_server_root*/profileTemplates/dmgr for the -templatePath parameter to create this type of profile. The dmgr type is the default profile when the -templatePath parameter is omitted.

Application Server profile

The basic function of the Application Server is to serve applications to the Internet or to an intranet.

An important product feature for the Network Deployment product is the ability to scale up a stand-alone Application Server profile by adding the Application Server node into a deployment

manager cell. Multiple Application Server processes in a cell can deploy an application that is in demand. You can also remove an Application Server node from a cell to return the node to the status of a stand-alone Application Server.

Each stand-alone Application Server has its own administrative console application, which you use to manage the Application Server. You can also use the wsadmin scripting facility to perform every function that is available in the administrative console application.

No node agent process is available for a stand-alone Application Server unless you decide to add the Application Server node to a deployment manager cell. Adding the Application Server to a cell is known as *federation*. Federation changes the stand-alone Application Server into a managed node. You use the administrative console of the deployment manager to manage the node. If you remove the node from the deployment manager cell, use the administrative console and the scripting interface of the stand-alone Application Server to manage the process.

Specify *app_server_root*/profileTemplates/default for the -templatePath parameter to create this type of profile.

Cell profile

The basic function of the cell profile is to serve applications to the Internet or to an intranet under the management of the deployment manager.

To create a cell profile using the **manageprofiles** command, you must create two individual profiles, the cell deployment manager profile and the cell node profile. Additionally, you can have only one cell deployment manager profile tied to the cell node profile and vice versa when you create a cell. After you create the initial cell profile, you can create custom profiles or stand alone profiles and federate them into the deployment manager.

Specify *app_server_root*/profileTemplates/cell for the -templatePath parameter to create the cell profile with the **manageprofiles** command.

Custom profile

The basic function of this profile that belongs to a deployment manager cell is to serve applications to the Internet or to an intranet under the management of the deployment manager.

The deployment manager changes a custom profile to a managed node by adding the node into the cell. The deployment manager also changes an Application Server into a managed node when you add an Application Server into a cell. When either node is added to a cell, the node becomes a managed node. The *nodeagent* process is then instantiated on the managed node. The node agent acts on behalf of the deployment manager to control Application Server processes on the managed node. The node agent can start or stop Application Servers, for example.

A deployment manager can create multiple Application Servers on a managed node so long as the node agent process is running. Processes on the managed node can include cluster members that the deployment manager uses to balance the workload for heavily used applications.

Use the administrative console of the deployment manager to control all of the nodes that the deployment manager manages. You can also use the wsadmin scripting facility of the deployment manager to control any of the managed nodes. A custom profile does not have its own administrative console or scripting interface. You cannot manage the node directly with the wsadmin scripting facility. You must use the administrative interface of the deployment manager to manage a managed node.

A custom profile does not include default applications or a default server as the Application Server profile does. A custom profile is an empty node. Add the node to the deployment manager cell. Then you can use the administrative interface of the deployment manager to customize the managed node, by creating clusters and Application Servers.

Specify *app_server_root*/profileTemplates/managed for the -templatePath parameter to create this type of profile.

Default profiles

Profiles use the concept of a default profile when more than one profile exists. The default profile is set to be the default target for scripts that do not specify a profile. Most scripts support the -profileName parameter which enables the script to act on a profile other than the default one.

The deployment manager profile, named dmgr, is created during installation. The default profile in the Network Deployment product is the stand-alone application server created during installation. The name of the profile is default.

Security policy for Application Server profiles

In environments where you plan to have multiple stand-alone Application Servers, the security policy of each Application Server profile is independent of the others. Changes to the security policy in one Application Server profile is not synchronized with the other profiles.

Installed file set

You decide where to install the files that define a profile.

The default location is in the *user_data_root*/profiles directory. You can change the location in a parameter when using the command line tool. For example, assume that you create two profiles on an iSeries system with host name devhost1.

You can specify a different directory, such as /home/QEJBSVR/profiles/myprofile , using the-profilePath parameter of the manageprofiles command:

```
manageprofiles
   -profileName myprofile
   -profilePath /home/QEJBSVR/profiles/myprofile
```

The following directories exist within a typical profile. This example assumes that a profile named AppSrv01 exists and was created in the default directory:

- *user_data_root*/profiles/AppSrv01/bin
- *user_data_root*/profiles/AppSrv01/config
- *user_data_root*/profiles/AppSrv01/configuration
- *user_data_root*/profiles/AppSrv01/etc
- *user_data_root*/profiles/AppSrv01/installableApps
- *user_data_root*/profiles/AppSrv01/installedApps
- *user_data_root*/profiles/AppSrv01/installedConnectors
- *user_data_root*/profiles/AppSrv01/logs
- *user_data_root*/profiles/AppSrv01/PolicyDirector
- *user_data_root*/profiles/AppSrv01/properties
- *user_data_root*/profiles/AppSrv01/samples
- *user_data_root*/profiles/AppSrv01/temp
- *user_data_root*/profiles/AppSrv01/wstemp

Different profile types might include different subdirectories. This list is not intended to be exhaustive or exclusive.

Profiles: required disk space

A minimum amount of space must be available in the directory where you create a profile.

An error can occur when you do not provide enough space to create a profile. Verify that you have, in addition to the minimum space required for a particular profile, an additional 40 MB of space. The 40 MB of space is used for log files and temporary files.

You must have 200 MB of available disk space in the directory where you create an Application Server profile.

You must have 30 MB of available disk space in the directory where you create a deployment manager profile.

You must have 10 MB of available disk space in the directory where you create a custom profile.

You must have 230 MB of available disk space in the directory where you create a cell profile that contains a federated Application Server profile and a deployment manager profile.

Both the **manageprofiles** command and the Profile Management tool can create a cell profile that has both a federated Application Server profile and a deployment manager profile. However, the Profile Management tool and the **manageprofiles** command create cell profiles differently. The differences are important to understand in terms of the available disk space needed to create the cell profiles. You can create a cell profile in one pass through the Profile Management tool. In this case you need 230 MB of available disk space to create the cell profile. However, to create a cell profile using the **manageprofiles** command that is equivalent to the cell profile that the Profile Management tool creates, you must create two individual profiles, the cell deployment manager profile and the cell node profile. The cell deployment manage profile requires 30 MB of available disk space, while the cell node profile requires 200 MB of available disk space.

Setting up and using the profile environment through commands

You can set up and use the profile environment through commands or the Profile Management tool. Use this example to set up and use the profile environment through commands.

This task assumes a basic familiarity with the **manageprofiles** command, other Application Server commands, and system commands.

Before you can create and use a profile, you must install WebSphere Application Server.

Perform the following steps to create and use the profile environment. This example deals with the profile environment of a stand-alone Application Server on the Windows platform.

1. Create the server profile from the original installation by using the *app_server_root*\bin\manageprofiles.bat script for the Windows platform. The script is *app_server_root*/bin/manageprofiles.sh on operating systems such as AIX or Linux. The script is *app_server_root*/bin/manageprofiles on the i5/OS platform.

   Assume that you create the profile by using the defaults. The following script is an example for creating an Application Server profile on the Windows platform:

   ```
   C:\Program Files\IBM\WebSphere\AppServer\bin\manageprofiles -create
   -templatePath C:\Program Files\IBM\WebSphere\AppServer\profileTemplates\default
   ```

   (The script is displayed on multiple lines for printing purposes.)
2. Change directories to the \bin directory of the new server profile.

   For example, issue the following command:

   ```
   cd Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\bin
   ```
3. Start server1 by changing directories to the *app_server_root*\bin directory of the original installation and issuing the **startServer** command.

   ```
   startServer server1 -profileName AppSrv01
   ```

4. Display a list of the ports assigned during profile creation.

   Open the `portdef.props` file in the `Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\properties` directory.
5. Open the administrative console for server1.

   The port for the administrative console is defined on the HTTP_TRANSPORT_ADMIN setting. If the value of the setting is `20003`, specify the following Web address in your browser:

   `http://`*`hostname_or_IP_address`*`:20003/ibm/console/`

You created an Application Server profile, started an Application Server, and accessed the administrative console through commands.

Deploy an application.

Creating default profiles

This topic discusses creation of default profiles for the i5/OS platform.

Default profiles are shipped with the product. A profile is a set of configuration files and installed applications that make up your WebSphere Application Server environment.

- Install the Web server plug-in feature without the core product feature to create a default remote HTTP profile.
- Install the application client feature without the core product feature to create the default application client profile.
- Install WebSphere Application Server to create the default WebSphere Application Server profile.
- Federate the default WebSphere Application Server profile into a Network Deployment cell to create a federated default WebSphere Application Server profile.
- Install WebSphere Application Server Network Deployment to create a deployment manager profile.

Default WebSphere Application Server profile

The default WebSphere Application Server profile provides the necessary configuration files for starting and managing the application server that it contains. This profile also provides the services and resources that are required to deploy and run enterprise applications.

The name of the default WebSphere Application Server profile is *default*. The default profile is contained under the following directory structure:

`user_data_root``/profiles/default`

This topic describes the cell, node, servers, applications and ports for the default profile.

General properties

Each profile contains the following general properties:

Profile name
default

Cell name
The host name of the iSeries server. For example, *MYISERIES*. The cell name is case-sensitive.

Node name
The host name of the iSeries server. For example, *MYISERIES*. The node name is case-sensitive.

Application server name
server1. The application server name is case-sensitive.

Pre-deployed applications

The *server1* application server includes these applications:

The administrative console application
Use the administrative console application to configure and manage the application server, enterprise applications, and other WebSphere Application Server resources.

The *DefaultApplication* example application
Use the DefaultApplication exmple which contains the snoop, hello, and hitcount examples, to quickly verify your application server configuration.

The installation verification application (*ivtApp*)
Use the ivtApp application to run the ivt Qshell script and verify the application server configuration.

Samples gallery
Use the Samples gallery application to access Web pages that describe the Samples that are provided with the products, and how to build, install, and run them.

PlantsByWebSphere sample
The PlantsByWebSphere Sample application depicts an online plants store.

Default ports

The default profile is configured to use the ports that are listed in Port number settings in WebSphere Application Server versions. To change any of these ports, run the chgwassvr script from Qshell:

```
app_server_root/bin/chgwassvr -server server1 options
```

where *options* specifies the parameters for the port that you want to change. For example, to change the administrative console port to 9091, run the following command:

```
app_server_root/bin/chgwassvr -server server1 -admin 9091
```

For more information about the script, see the chgwassvr command in the *Using the administrative clients* PDF . Potential port conflicts with other versions or editions of WebSphere Application Server are noted where appropriate.

To display information such as node name, ports used, servers, and installed applications for the default profile, run the dspwasinst script from Qshell. For more information, see dspwasinst command in the *Using the administrative clients* PDF.

Default application client profile

The application client profile contains configuration files and properties files that are used to configure and run an application client. This profile does not contain an application server.

The name of the default application client profile is *client*. This profile is located in the following directory:

```
user_data_root/profiles/client
```

The application client profile is created when you install only the Application client feature or when you install the Application client feature with the Web server plug-ins feature. This profile is not created when you install the Core Product Option feature.

In an application client configuration, the application client and the application server can run on separate machines or logical partitions.

Federated default WebSphere Application Server profile

You can use the addNode script or the Network Deployment administrative console to federate a profile into a Network Deployment cell.

When you federate a profile into a cell, a node agent server is created to monitor the application server. The node agent server serves as an intermediary between the application servers on the node and the deployment manager that oversees the entire cell.

For more information on working with federated profiles, see the *Administering applications and their environment* PDF.

This article describes the default profile that is provided with WebSphere Application Server after it is federated, or added, to a Network Deployment cell.

- "General properties"
- "Predeployed applications"
- "Default ports"

General properties

The following properties are available for the default profile:

Profile name
default

Cell name
The cell name with the form *host_name*Network, where *host_name* is the host name of the iSeries server. For example, *MYISERIESNetwork*. The cell name is case-sensitive.

Node name
The host name of the iSeries server. For example, *MYISERIES*. The node name is case-sensitive.

Application server name
server1. The application server name is case-sensitive.

Node agent server name
nodeagent. The server name of the node agent is case-sensitive.

Predeployed applications

The *server1* application server includes these applications:

The *DefaultApplication* example application
Use DefaultApplication application, which contains the snoop, hello, and hitcount examples that quickly verify your application server configuration.

The install verification application (*ivtApp*)
Use *ivtApp* application to use the ivt Qshell script and verify the application server configuration.

Samples gallery
Use Samples gallery application to access Web pages that describe the samples that are provided with the products, and how to build, install, and run them.

PlantsByWebSphere sample
The *PlantsByWebSphere* sample application depicts an online plants store.

Default ports

The default profile is configured to use the ports that are listed in Port number settings in WebSphere Application Server versions. To change any of these ports, use the deployment manager administrative

console or run the chgwassvr script from Qshell and specify the deployment manager profile on the -profileName option. The chgwassvr script is in the *app_server_root*/bin directory.

For more information about the script, see the chgwassvr command topic in the *Using the administrative clients* PDF. Potential port conflicts with other versions or editions of WebSphere Application Server are noted where appropriate.

To display information such as node name, ports used, servers, and installed applications for the default profile, run the dspwasinst script from Qshell. For more information, see the dspwasinst command in the *Using the administrative clients* PDF.

Network Deployment deployment manager profile

When you install WebSphere Application Server Network Deployment, two profiles are created: a stand-alone application server profile named *default*; and a deployment manager profile named *dmgr*. The stand-alone application server profile is the default profile with respect to scripts.

For information about the default stand-alone application server profile, see "Default WebSphere Application Server profile" on page 126. When invoking scripts which act upon, or are relative to, the deployment manager, you must specify `-profileName dmgr` when invoking the script.

The name of the deployment manager profile that is created when you install Network Deployment is `dmgr`. The deployment manager profile is contained in the directory structure under

user_data_root/profiles/dmgr

This profile provides the necessary configuration files for starting and managing the deployment manager server that it contains. The profile also provides everything necessary to configure and manage WebSphere Application Server profiles, or nodes, that are in the deployment manager cell.

The deployment manager profile contains an application server with a server name of *dmgr*. The dmgr application server is a special application server that contains the deployment manager. The dmgr server contains the Network Deployment administrative console application and the Network Deployment file transfer application. These applications enable the distributed management of one or more WebSphere Application Server profiles, or nodes.

General properties

The following properties are available for the default profile:

Profile name
dmgr.

Cell name
*hostname*Network, where *hostname* is the host name of the iSeries server. For example, MYISERIESNetwork. The cell name is case-sensitive.

Node name
*hostname*Manager, where *hostname* is the host name of the iSeries server. For example, MYISERIESManager. The node name is case-sensitive.

Application server name
dmgr. The application server name is case-sensitive.

Predeployed applications

The dmgr server includes these applications, which enable the distributed management of one or more WebSphere Application Server profiles, or nodes.

The Network Deployment administrative console application
Use administrative console application to configure and manage the deployment manager, application servers, enterprise applications, and other WebSphere Application Server resources within the Network Deployment cell.

The filetransfer application
The file transfer application transfers configuration and application files from the deployment manager to one or more managed nodes in the Network Deployment cell.

Default ports

The deployment manager profile is configured to use the ports that are listed in “Port number settings in WebSphere Application Server versions” on page 1. To change any of these ports, run the chgwassvr script from Qshell:

```
app_server_root/bin/chgwassvr
 -server dmgr [options]
```

where [options] specifies the parameters for the port you want to change. For example, to change the administrative console port to 9091, run this command:

```
app_server_root/bin/chgwassvr
 -server dmgr -admin 9091
```

The command is split on multiple lines for printing purposes.

For more information about the script, see chgwassvr command.

To display information such as node name, ports used, servers and installed applicatio6ns for the default profile, run the dspwasinst script from Qshell. For more information, see dspwasinst command.

Default remote HTTP profile

The name of the default remote HTTP profile is *http*. This profile is created when you install only the Web server plug-in feature or when you install the Web server plug-in feature with the Application client feature. The profile is not created when you install the Core product feature.

The remote profile is contained under the following directory structure:

```
user_data_root/profiles/http
```

The remote profile does not contain an application server, but rather contains configuration files and log files to configure a remote HTTP topology. In a remote HTTP configuration, the HTTP server and the application server run on separate machines or logical partitions. For more information, see “Selecting a Web server topology diagram and roadmap” on page 12.

Deleting a profile

This topic describes how to manually delete a profile.

If a node within a profile is federated to a deployment manager, before you delete the profile, stop the node and remove the node from the deployment manager. Otherwise, an orphan node is left in the deployment manager.

Before using the manual procedure to remove a profile, try the **manageprofiles** command with the -delete option. For example, issue the following command for your operating system platform:

```
./manageprofiles -delete
                 -profileName profile_name
```

If you delete a profile that has augmenting templates registered to it in the profile registry, then unaugment actions are attempted prior to the deletion.

If the command does not work, use this procedure to delete the profile.

This procedure describes how to manually delete a profile when the **manageprofiles -delete** command results in the following message:

```
INSTCONFFAILED: Cannot delete profile.
```

1. Issue operating system commands to delete the profile directory.
2. Issue the following command to remove references in the registry to deleted profiles:

   ```
   manageprofiles -validateAndUpdateRegistry
   ```

   Editing of the registry is not recommended.

You have now deleted a profile.

See the description of the the manageprofiles command topic in the *Using the administrative clients* PDF to learn more about the command-line method of working with profiles.

Chapter 4. Setting up the administrative architecture

You can monitor and control incorporated nodes and the resources on those nodes by using these tasks with the administrative console or other administrative tools.

1. Use the settings page for an administrative service to configure administrative services.
2. Configure cells.
3. Configure deployment managers.
4. Manage nodes.
5. Manage node agents.
6. Manage node groups.
7. Configure remote file services.

Cells

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

A cell is a configuration concept, a way for administrators to logically associate nodes with one another. Administrators define the nodes that make up a cell, according to the specific criteria that make sense in their organizational environments.

Administrative configuration data is stored in XML files. A cell retains master configuration files for each server in every node in the cell. Each node and server also have their own local configuration files. Changes to a local node or to a server configuration file are temporary, if the server belongs to the cell. While in effect, local changes override cell configurations. Changes to the master server and master node configuration files made at the cell level replace any temporary changes made at the node when the cell configuration documents are synchronized to the nodes. Synchronization occurs at designated events, such as when a server starts.

Configuring cells

This topic describes how to change the cell protocol information, define custom properties for the cell, and add additional nodes.

Before you can configure cells, you must install the WebSphere Application Server Network Deployment product.

When you create a deployment manager profile, a cell is created. A cell provides a way to group one or more nodes of your Network Deployment product. You probably do not need to configure the cell again. To view information about and to manage a cell, use the settings page for a cell.

1. Access the settings page for a cell. Click **System Administration > Cell** from the navigation tree of the administrative console.
2. If the protocol that the cell uses to retrieve information from a network is not appropriate for your system, select the appropriate protocol. By default, a cell uses Transmission Control Protocol (TCP). If you want the cell to use User Datagram Protocol, select **UDP** from the list for **Cell Discovery Protocol** on the settings page for the cell. It is unlikely that you need to change the cell protocol configuration from TCP.
3. Click **Custom Properties** and define any name-value pairs that your deployment manager needs.
4. When you install the WebSphere Application Server Network Deployment product, a node is added to the cell. You can add additional nodes on the Node page. Click **Nodes** to access the Node page, which you use to manage nodes.

WebSphere Application Server on other platforms has support for Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6). However, the i5/OS platform supports IPv4 only. When you add a node to a cell, the format in which you specify the host name is based on the version of IP that the node is using. For details, see “IP version considerations for cells.”

Depending on which steps you performed, you changed the cell protocol information, defined custom properties for the cell, and added additional nodes.

You can continue to administer your Network Deployment product by doing such tasks as managing nodes, node agents, and node groups.

IP version considerations for cells

There are compatibility issues to consider when configuring the IP version for cells.

WebSphere Application Server has support for Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6). However, on the i5/OS platform, WebSphere Application Server V6 supports IPv4 only.

When defining a node, you must specify the host name as a string or as a 32-bit numerical address.

Specifying host names

When you create a new profile, you can optionally specify the host name for the profile by using the -hostName parameter. If you do not specify the host name, the host name for the profile defaults to the fully qualified host name for the iSeries server. If you choose to use the -hostName parameter and want to specify the IP address for your iSeries server, specify the valid 32-bit numerical address.

Multicast configuration

WebSphere Application Server uses multicast broadcasting at the node level to allow a node agent to discover the managed processes in the node. IPv4 and IPv6 addresses are not compatible. Therefore, to allow a WebSphere Application Server node to run after initial installation, both IPv4 and IPv6 multicast addresses are initially defined in the node agent configuration, and when a node agent starts, both addresses are tried in sequence. Delete the NODE_IPV6_MULTICAST_DISCOVERY_ADDRESS address after installation because, because limiting multicast discovery to the known protocol, the node agent runs more efficiently.

Deleting the Internet Protocol Version 4 or the Internet Protocol Version 6 multicast port

This topic describes how to delete the Internet Protocol Version 4 (IPv4) or the Internet Protocol Version 6 (IPv6) for multicast ports so that the node agent runs more efficiently.

You must install the WebSphere Application Server Network Deployment product before you can delete a multicast port.

To allow node installation to run out-of-the box, both IPv4 and IPv6 are initially defined in the node agent configuration. To make the node agent run more efficiently, delete the multicast port that the node is not using.

To delete the IPv6 multicast port using the Administrative Console, perform these steps:

1. Click **System Administration > Node Agents**.
2. Select the node agent.
3. On the next panel, under Additional Properties, select **Ports**.

   The next panel shows a list of existing ports.

4. Select NODE_IPV6_MULTICAST_DISCOVERY_ADDRESS to delete IPv6.
5. Click **Delete**.

You deleted IPv6.

You can continue to administer your Network Deployment product by doing such tasks as managing nodes, node agents, and node groups.

Cell settings

Use this page to set the discovery protocol and address end point for an existing cell. A cell is a configuration concept, a way for an administrator to logically associate nodes according to whatever criteria make sense in the administrator's organizational environment.

To view this administrative console page, click **System Administration > Cell**.

Name

Specifies the name of the existing cell.

A cell name must be unique in any circumstance in which the product is running on the same physical machine or cluster of machines, such as a sysplex. Additionally, a cell name must be unique in any circumstance in which network connectivity between entities is required either between the cells or from a client that must communicate with each of the cells. Cell names also must be unique if their name spaces are going to be federated. Otherwise, you might encounter symptoms such as a javax.naming.NameNotFoundException exception, in which case, you need to create uniquely named cells.

Short Name

Specifies the short name of the cell. The name is 1-8 characters, alphanumeric or national language. It cannot start with a numeric.

The short name property is read only. It was defined during installation and customization.

Cell Discovery Protocol

Specifies the protocol that the cell follows to retrieve information from a network.

Select one of these protocol options:
UDP User Datagram Protocol (UDP)
TCP Transmission Control Protocol (TCP)

Default TCP

Starting the WebSphere Application Server Network Deployment environment

WebSphere Application Server Network Deployment profiles run in the QWAS61 subsystem and require that TCP/IP is configured and activated on the system. This topic discusses how to prepare your system to run WebSphere Application Server Network Deployment profiles.

TCP/IP must be active before the WebSphere Application Server subsystem can start. Ensure that the STRTCP command runs before the STRSBS QWAS61/QWAS61 command in your startup program or in your autostart job.

The system startup program is defined by the QSTRUPPGM system value. For more information about the QSTRUPPGM system value, see the *Work Management Guide* (SC41-5306) (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/QB3ALG03/CCONTENTS).

Complete this task to configure your system to

- Run WebSphere Application Server Network Deployment profiles
- Configure these profiles to start automatically when the QWAS61 subsystem starts
- To prepare your system to run WebSphere Application Server Network Deployment profiles, follow these steps.
  1. Start Transmission Control Protocol/Internet Protocol (TCP/IP). On the i5/OS command line, enter this command:

     ```
     STRTCP
     ```

  2. To start your application server profile, see one of these topics:
     - "Starting an application server" on page 221
     - "Starting and stopping the deployment manager" on page 138

     Note:
     - Unlike previous releases, releases starting with WebSphere Application Server V6 do not start the default application server when the subsystem is started. When you use the startServer or startManager Qshell script to start your server profile, the QWAS61 subsystem is started if it is not currently active.
     - WebSphere Application Server Network Deployment includes support for stand-alone application server profiles, managed (federated) profiles, and deployment manager profiles without requiring the WebSphere Application Server product to also be installed. When you install Network Deployment, two profiles are created: the default profile and the dmgr profile. The default profile is a stand-alone application server profile and the dmgr profile is a deployment manager profile.
  3. Use one of these methods to verify that your server is running:
     - On the i5/OS command line, run the Work with Active Jobs (WRKACTJOB) command:

       ```
       WRKACTJOB SBS(QWAS61)
       ```

       Your server job is listed. When the server is ready to accept requests, the joblog contains message WAS0106: `WebSphere application server` *`serverName`* `ready`.
     - In Operations Navigator, look for QIBM_WSA_ADMIN under **Server Jobs**.
- If your profile has authority to the QWAS61/QWASJOBD job description and QWAS61/QWAS61 subsystem description, you can configure WebSphere Application Server Network Deployment profiles to automatically start when the QWAS61 subsystem starts. Perform these steps for each profile:
  1. Create a duplicate of the job description that is used by WebSphere Application Server Network Deployment profiles. For example, on the i5/OS command line, run this command:

     ```
     CRTDUPOBJ OBJ(QWASJOBD) FROMLIB(QWAS61) OBJTYPE(*JOBD) TOLIB(mywasjobd) NEWOBJ(mydmgr)
     ```

  2. Use the CHGJOBD command to change the newly created job description so that the Request data or command (RQSDTA) field starts the server. For example, to start the default deployment manager server (dmgr), set the RQSDTA value as follows:

     ```
     'QSYS/CALL PGM(QWAS61/QWASSTRSVR)
     PARM(''-profilePath''      ''/QIBM/UserData/WebSphere/
     AppServer/V61/ND/profiles/
     dmgr''     ''-server'' ''dmgr'')'
     ```

     To start the default application server (server1), set the RQSDTA value as follows:

     ```
     'QSYS/CALL PGM(QWAS61/QWASSTRSVR) PARM(''-profilePath''
         ''/QIBM/UserData/WebSphere/AppServer/V61/ND/profiles/default''
         ''-server'' ''server1'')'
     ```

3. Add an autostart job entry to the QWAS61/QWAS61 subsystem. Enter this command from the i5/OS command line:

   ```
   ADDAJE SBSD(QWAS61/QWAS61) JOB(mydmgr) JOBD(mywasjobd/mydmgr)
   ```

   You can configure the system so that the QWAS61 subsystem starts at iSeries system startup. To enable automatic startup, add the following line to the system startup program:

   ```
   STRSBS QWAS61/QWAS61
   ```

You have configured your system to run WebSphere Application Server Network Deployment profiles, and might have configured these profiles to start automatically when the QWAS61 subsystem starts.

Deploy an application to get started.

Deployment managers

Deployment managers are administrative agents that provide a centralized management view for all nodes in a cell, as well as management of clusters and workload balancing of application servers across one or several nodes in some editions.

A deployment manager hosts the administrative console. A deployment manager provides a single, central point of administrative control for all elements of the entire WebSphere Application Server distributed cell. Each cell contains one deployment manager.

Configuring deployment managers

Configure deployment managers for a single, central point of administrative control for all elements in a WebSphere Application Server distributed cell.

When you created a deployment manager profile, a deployment manager was created. You can run the deployment manager with its default settings. However, you can follow this task to change the deployment manager configuration settings such as the ports the process uses, custom services, logging and tracing settings, and so on. To view information about and manage a deployment manager, use the settings page for a deployment manager.

1. Access the settings page for a deployment manager. Click **System Administration > Deployment Manager** from the navigation tree of the administrative console.
2. Configure the deployment manager as desired by clicking on a property such as **Custom Services** and specifying settings on the resulting pages.

Deployment manager settings

Use this page to stop the deployment manager from running, and to link to other pages which you can use to define additional properties for the deployment manager. A deployment manager provides a single, central point of administrative control for all elements of the entire WebSphere Application Server distributed cell.

To view this administrative console page, click **System administration > Deployment manager**.

Name

Specifies a logical name for the deployment manager. The name must be unique within the cell.

Data type String

Short name

Specifies the short name of the deployment manager 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.

Data type String

Unique Id
Specifies the unique ID of this deployment manager server.

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

Data type String

Process ID
Specifies a string identifying the process.

Data type String
Default None

Cell Name
Specifies the name of the cell for the deployment manager. The default is the name of the host computer on which the deployment manager is installed with `Cell##` appended, where *##* is a two-digit number.

Data type String
Default *host_name*Cell01

Node Name
Specifies the name of the node for the deployment manager. The default is the name of the host computer on which the deployment manager is installed with `CellManager##` appended, where *##* is a two-digit number.

Data type String
Default *host_name*CellManager01

State
Indicates the state of the deployment manager. The state is *Started* when the deployment manager is running and *Stopped* when it is not running.

Data type String
Default Started

Starting and stopping the deployment manager

The deployment manager is an administration application that runs in a special application server, which is created when you install WebSphere Application Server Network Deployment or when you create a new profile using the deployment manager profile template. With the deployment manager, you can administer multiple WebSphere Application Server nodes. This topic describes how you start and stop the application server for the deployment manager.

Before you can start or stop the deployment manager, you must first install WebSphere Application Server Network Deployment.

Start the deployment manager so that you can manage all the elements of the WebSphere Application Serve cell. Stop the deployment manager as needed, such as when migrating to a new version of WebSphere Application Server Network Deployment, when uninstalling the product, and so on.

- Start the deployment manager.

  Use one of these methods to start the application server for a deployment manager:

  – Use the startManager Qshell script .

    For information about how to start a deployment manager application server with the startManager script, see the startManager command topic in the *Administering applications and their environment* PDF.

  – Use the startServer Qshell script.

    For information about how to start an application server with the startServer script, see the startServer command topic in the *Using the administrative clients* PDF.

  – Use the Submit Job (SBMJOB) CL command.

    You can run this CL command from an i5/OS command line:

    ```
    SBMJOB CMD(CALL PGM(QWAS61/QWASSTRSVR) PARM('-profilePath'
      'profile_root' '-server' 'server')) JOB(server)
      JOBD(QWAS61/QWASJOBD) JOBQ(QWAS61/QWASJOBQ) USER(QEJBSVR) LANGID(*USRPRF)
      CNTRYID(*USRPRF) CCSID(*USRPRF) OUTQ(QWAS61/QWASOUTQ)
    ```

    where *profile_root* is the fully qualified path of the directory that contains your profile and *server* is the name of the server in that profile.

- Stop the deployment manager.

  Use one of these methods to stop the application server for a deployment manager:

  – Use the stopManager Qshell script.

    For information about how to stop the deployment manager with the stopManager script, see the stopManager command topic in the *Using the administrative clients* PDF.

  – Use the stopServer Qshell script.

    For information about how to stop the deployment manager with the stopServer script, see the stopServer command topic in the *Using the administrative clients* PDF.

  – Use the End Job (ENDJOB) CL command.

    To use the ENDJOB CL command to end an application server, enter this command on an i5/OS command line:

    ```
    ENDJOB JOB(jobNumber/QEJBSVR/jobName) OPTION(*CNTRLD) DELAY(delayTime)
    ```

    where *jobNumber* is the job number, *jobName* is the name of the application server job for the deployment manager, and *delayTime* is the amount of time to wait for the job to end in seconds. Use a value of 600 seconds initially. To find out what the appropriate delayTime is, see “Shutting down the WebSphere Application Server subsystem” on page 207.

  – Use the Network Deployment deployment manager administrative console.

    To stop the deployment manager from the administrative console:

    1. Start the administrative console. For more information, see the Starting the administrative console for the deployment manager topic in the *Installing your application serving environment* PDF.
    2. Expand **System Administration** and click **Deployment Manager**.
    3. Click **Stop** on the Configuration sheet.

You have started the deployment manager and have optionally stopped it.

Deploy an application.

Node

A *node* is a logical grouping of managed servers.

A node usually corresponds to a logical or physical computer system with a distinct IP host address. Nodes cannot span multiple computers. The node name for the default profile that is created during product installation is the short host name of the iSeries server. When you create a new profile that is not a deployment manager profile, the node name defaults to *hostName_profileName*.

Nodes in the network deployment topology can be managed or unmanaged. A managed node has a node agent process that manages its configuration and servers. Unmanaged nodes do not have a node agent.

A managed node has a node agent that manages all servers on a node, whether the servers are WebSphere Application Servers, Java Message Service (JMS) servers (on Version 5 nodes only), Web servers, or generic servers. The node agent represents the node in the management cell and keeps the configuration up to date.

An unmanaged node does not have a node agent to manage its servers. Unmanaged nodes in the Network Deployment environment can have server definitions such as Web servers, but not Application Server definitions. Unmanaged nodes in the Network Deployment environment cannot have a node agent added to it, and therefore cannot become a managed node. In the stand-alone Application Server environment, nodes do not have node agents and are also considered unmanaged nodes. The deployment manager cannot manage a stand-alone Application Server because it is not known to the cell. A stand-alone Application Server can be federated. When it is federated, a node agent is automatically created, and the node becomes a managed node in the cell.

A supported Web server can be on a managed node or an unmanaged node. You can define only one Web server to a stand-alone WebSphere Application Server node. This Web server is defined on an unmanaged node. You can define Web servers to the deployment manager. These Web servers can be defined on managed or unmanaged nodes.

WebSphere Application Server supports basic administrative functions for all supported Web servers. For example, the generation of a plug-in configuration can be performed for all Web servers. However, propagation of a plug-in configuration to remote Web servers is supported only for IBM HTTP Servers that are defined on an unmanaged node. If the Web server is defined on a managed node, propagation of the plug-in configuration is done for all the Web servers by using node synchronization. The Web server plug-in configuration file is created according to the Web server definition and is based on the list of applications that are deployed on the Web server. You can also map all supported Web servers as potential targets for the modules during application deployment.

WebSphere Application Server supports some additional administrative console tasks for IBM HTTP Servers on managed and unmanaged nodes. For instance, you can start IBM HTTP Servers, stop them, terminate them, display their log files, and edit their configuration files.

You can add managed and unmanaged nodes to a Network Deployment cell in one of the following ways:

- Administrative console
- Command line (managed nodes only)
- Administrative script
- Java program

Each of these methods for adding a managed node to a Network Deployment cell includes the option of specifying a target node group for the managed node to join. If you do not specify a node group, or you do not have the option of specifying a node group, the default node group of DefaultNodeGroup is the target node group.

Whether you specify an explicit node group or accept the default, the node group membership rules must be satisfied. If the node that you are adding does not satisfy the node group membership rules for the target node group, the add node operation fails with an error message.

Managing nodes

This topic describes how to add a node, select the discovery protocol for a node, define a custom property for a node, stop servers on a node, and remove a node.

A node is a grouping of managed or unmanaged servers. You can add both managed and unmanaged nodes to the WebSphere Application Server topology. If you add a new node for an existing WebSphere Application Server to the Network Deployment cell, you add a managed node. If you create a new node in the topology for managing Web servers or servers other than WebSphere Application Servers, you add an unmanaged node.

To view information about nodes and managed nodes, use the Nodes page. To access the Nodes page, click **System Administration > Nodes** in the administrative console navigation tree.

You can manage nodes on an application server through the wsadmin scripting tool, through the Java application programming interfaces (APIs), or through the administrative console. Perform the following tasks to manage nodes on an application server through the administrative console.

- **Add a node.**
  1. Go to the Nodes page and click **Add Node**. Choose whether you want to add a managed or unmanaged node, and click **Next**.
  2. For a managed node, verify that an application server is running on the host for the node that you are adding. On the Add Node page, specify a host name, connector type, and port for the application server at the node you are adding.
  3. For a managed node, perform one of the following sets of actions listed in the table:

If the deployment manager is on	And the node that you add to the cell is on	Complete the appropriate set of actions:
The distributed platform or the i5/OS platform	The distributed platform or the i5/OS platform	Optionally specify a node group and a core group. Click **OK**.
The distributed platform or the i5/OS platform	A z/OS system	Specify a node group that contains nodes from the same sysplex as the node you are now adding. If no such node group exists, create a node group and then specify that node group. Optionally specify a core group. Click **OK**.
A z/OS system	The distributed platform or the i5/OS platform	Specify a node group that contains distributed nodes. If no such node group exists, create a node group and then specify that node group. Optionally specify a core group. Click **OK**.

For the node group option to display, a group other than the default node group must first be created. Likewise, for the core group option to display, a group other than the default core group must first be created.

4. For an unmanaged node, on the **Nodes > New** page, specify a node name, a host name, and a platform for the new node. Click **OK**.

The node is added to the WebSphere Application Server environment and the name of the node is displayed in the collection on the Nodes page.

Both Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) are now supported by WebSphere Application Server, but restrictions do apply when using both IPv4 and IPv6 in the same cell. When you add a node to a cell, the format in which you specify the name is based on the version of IP that the node is using. For details, see IP version considerations for cells.

- **Select the discovery protocol.**

  If the discovery protocol that a node uses is not appropriate for the node, select the appropriate protocol. On the Nodes page, click the node to access the Settings for the node. Select a value for **Discovery Protocol**. User Datagram Protocol (UDP) is faster than Transmission Control Protocol (TCP). However, TCP is more reliable than UDP because UDP does not guarantee the delivery of datagrams to the destination. The default of TCP is the recommended value.

  For a node agent or deployment manager, use **TCP** or **UDP**.

  A managed process uses multicast as its discovery protocol. The discovery protocol is fixed for a managed process. The main benefit of using multicast on managed processes is efficiency for the node agent. Suppose you have forty servers in a node. A node agent that uses multicast sends one broadcast to all forty servers. If a node agent did not use multicast, it would send discovery queries to all managed processes one at a time, totaling forty sends. Additional benefits of using multicast are that you do not have to configure the discovery port for each server or prevent port conflicts because all servers in one node listen to one port instead of to one port for each server.

- **Define a custom property for a node.**
  1. On the Nodes page, click the node for which you want to define a custom property.
  2. On the Settings for the node, click **Custom Properties**.
  3. On the Property collection page, click **New**.
  4. On the Settings page for a property instance, specify a name-value pair and a description for the property, and click **OK**.

- Synchronize the node configuration.

  If you add a managed node or change a managed node configuration, synchronize the node configuration. On the Node Agents page, ensure that the node agent for the node is running. Then, on the Nodes page, select the check box beside the node whose configuration files you want to synchronize and click **Synchronize** or **Full Resynchronize**.

  Clicking either option sends a request to the node agent for that node to perform a configuration synchronization immediately, instead of waiting for the periodic synchronization to occur. This action is important if automatic configuration synchronization is disabled, or if the synchronization interval is set to a long time, and a configuration change is made to the cell repository that needs to replicate to that node. Settings for automatic synchronization are on the File Synchronization Service page.

  Synchronize requests that a node synchronization operation be performed using the normal synchronization optimization algorithm. This operation is fast, but might not fix problems from manual file edits that occur on the node. It is still possible for the node and cell configuration to be out of synchronization after this operation is performed.

  Full Resynchronize clears all synchronization optimization settings and performs configuration synchronization anew, so there is no mismatch between node and cell configuration after this operation is performed. This operation can take longer than the **Synchronize** operation.

  Unmanaged nodes cannot be synchronized.

- **Stop servers on a node.**

  On the Nodes page, Select the check box beside the managed node whose servers that you want to stop running, and click **Stop**.

- **Remove a node.**

On the Nodes page, Select the check box beside the node that you want to delete and click **Remove Node**. If you cannot remove the node by clicking **Remove Node**, remove the node from the configuration by clicking **Force Delete**.

- **View node capabilities.**

  Review the node capabilities, such as the product version through the administrative console. You can also query them through the Application Server application programming interface (API) or the wsadmin tool. For information on the wsadmin tool, see the *Using the administrative clients* PDF.

  The product versions for WebSphere Application Server are as follows: The base edition of WebSphere Application Server is listed in the version column as `Base`. The express edition of WebSphere Application Server is listed in the version column as `Express`. The Network Deployment product is listed in the version column as `ND`.

Node collection

Use this page to manage nodes in the WebSphere Application Server environment. Nodes group managed servers. The table lists the managed and unmanaged nodes in this cell. The first node is the deployment manager. Add new nodes to the cell and to the list by clicking **Add Node**.

To view this administrative console page, click **System administration > Nodes**.

Name

Specifies a name for a node that is unique within the cell.

A node corresponds to a physical computer system with a distinct IP host address. The node name is usually the same as the host name for the computer.

Both Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) are now supported by WebSphere Application Server, but there are restrictions that apply to using both IPv4 and IPv6 in the same cell. Note that when you add a node to a cell, the format in which you specify the name is based on the version of IP the node will be using.

Version

Specifies the product version of the node.

The product version is the version of a WebSphere Application Server for managed nodes. For unmanaged nodes on which you can define Web servers, the version displays as not applicable

The base edition of WebSphere Application Server is listed in the version column as `Base`. The express edition of WebSphere Application Server is listed in the version column as `Express`. The Network Deployment product is listed in the version column as `ND`.

Discovery protocol

Specifies the protocol that servers use to discover the presence of other servers on this node.

The possible protocol options follow:
UDP User Datagram Protocol (UDP)
TCP Transmission Control Protocol (TCP)

Status

Indicates that the node is either synchronized, not synchronized, unknown, or not applicable.

Status	Explanation
Synchronized	The configuration files on this node are synchronized with the deployment manager.

Status	Explanation
Not synchronized	The configuration files on this node are not synchronized with the deployment manager and are out-of-date. Perform a synchronize operation to get the latest configuration changes on the node.
Unknown	The state of the configuration file cannot be determined because the node agent cannot be reached for this node.
Not applicable	The status column is not applicable for this node because the node is an unmanaged node.

Node settings

Use this page to view or change the configuration or topology settings for either a managed node instance or an unmanaged node instance.

A managed node is a node with an Application Server and a node agent that belongs to a cell. An unmanaged node is a node defined in the cell topology that does not have a node agent running to manage the process. Unmanaged nodes are typically used to manage Web servers.

To view this administrative console page, click **System administration > Nodes >** *node_name*.

Name:

Specifies a logical name for the node. The name must be unique within the cell.

A node name usually is identical to the host name for the computer. However, you choose the node name. You can make the node name some name other than the host name.

Data type String

Both Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) are now supported by WebSphere Application Server, but there are restrictions that apply to using both IPv4 and IPv6 in the same cell. Note that when you add a node to a cell, the format in which you specify the name is based on the version of IP the node will be using.

Short Name:

Specifies the name of a node. The name is 1-8 characters, alphanumeric or national language. It cannot start with a numeric.

The short name property is defined during installation and customization. However, you can change the short name using the **renameNode.sh** command.

Host name:

Specifies the host name of the unmanaged node that is added to the configuration.

Date type String
Default None

Discovery Protocol:

Specifies the protocol that the node follows to retrieve information from a network. The Discovery protocol setting is only valid for managed nodes.

Select from one of these protocol options:
UDP User Datagram Protocol (UDP)
TCP Transmission Control Protocol (TCP)

Data type	String
Default	TCP
Range	Valid values are UDP or, TCP.

UDP is faster than TCP, but TCP is more reliable than UDP because UDP does not guarantee delivery of datagrams to the destination. Between these two protocols, the default of TCP is recommended.

File permissions: Specifies the most lenient file permissions for the application files that WebSphere Application Server extracts into the application destination location. A deployer can override the permissions by configuring the permissions at the application level. However, if the file permissions specified at the application level are more lenient than the ones specified at the node, the ones specified at the node are used. The File permissions setting is only valid for managed nodes.

Data type	String
Default	`755` , or `rwx-rx-rx`, for files that end in `.dll`, `.so`, `.a` and `.sl` if no value is set

Platform type:

Specifies the operating system on which the unmanaged node runs.

Valid options are:

Windows
AIX
HP-UX
Solaris
Linux
OS/400
z/OS

Add managed nodes

A managed node is a node with an Application Server and a node agent that belongs to a cell. Use this page to add a managed node to a cell.

To view this administrative console page, click **System Administration > Nodes > Add node > Next** .

Node connection

Specifies connection information for WebSphere Application Server.

- **Host**

  Specifies the host name or IP address of the node to add to the cell. A WebSphere Application Server instance must be running on this machine.

Data type	String
Default	None

- **JMX connector type**

  Specifies the Java Management Extensions (JMX) connectors that communicate with the WebSphere Application Server when you invoke a scripting process.

Select from one of these JMX connector types:

Simple Object Access Protocol (SOAP)

Use when the Application Server connects to a SOAP server.
Remote Method Invocation (RMI)

Use when the Application Server connects to an RMI server.

- **JMX connector port**

  Specifies the port number of the JMX connector on the instance to add to the cell. The default SOAP connector port is 8880.

Date type	Integer
Default	8880

- **Application server user name**

  Specifies the administration user name that connects to the remote Application Server whose node is being added to the cell. The Application Server user name and password are used to connect to the Application Server and start the add node process at the Application Server. The Application Server user name and password settings always display. You must specify values for them if security is enabled at the Application Server. Otherwise, leave them blank. User name requirements are the requirements that the security system that you use imposes for a user name.
- **Application server password**

  Specifies the password for the Application Server user name that you supply. Password requirements are the requirements that the security system that you use imposes for a password.
- **Deployment manager user name**

  Specifies the deployment manager administration user name that the Application Server uses when connecting to the deployment manager to add its node to the cell. The deployment manager user name and password settings display only if security is enabled at the deployment manager. The deployment manager user name and password are required if their settings display. User name requirements are the requirements that the security system that you use imposes for a user name.
- **Deployment manager password**

  Specifies the password for the deployment manager user name that you supply. Password requirements are the requirements that the security system that you use imposes for a password.

Options

Select from the following settings to further specify characteristics when adding a managed node to a cell.

- **Include Applications**

  Copies the applications installed on the remote instance into a cell. If the applications to copy have the same name as the applications that currently exist in the cell, the Application Server does not copy the applications.
- **Include buses**

  Specifies whether to move the bus configuration at the node to the deployment manager.
- **Starting port**

  Specifies the port numbers for the node agent process.

Use default	Specifies whether to use the default node agent port numbers.

Specify	Allows you to specify the starting port number in the Port number field. WebSphere Application Server administration assigns the port numbers in order from the starting port number. For example, if you specify 9950, the administration program configures the node agent ports as 9950, 9951, 9952, and so on.

- **Core Group**

  Specifies the group to which you can add a cluster or node agent. By default, clusters or node agents are added to the DefaultCoreGroup group.

  Select from one of the core groups if a list is displayed. The list displays if a core group in addition to the default core group exists.
- **Node group**

  Specifies the group to which you can add the node. By default, nodes are added to the DefaultNodeGroup group.

  Select from one of the node groups if a list is displayed. The list displays if a node group in addition to the default node group exists.

Node installation properties

Use this page to view read-only installation properties for this node. These properties provide information about the capabilities of the node that are collected during product installation time, such as the operating system name, architecture and version, or WebSphere Application Server product levels that are installed on the node.

To view this administrative console page, click **System administration > Nodes > *node name* > Node installation properties**.

Information about a node, such as operating system platform and product features, is maintained in the configuration repository in the form of properties. As product features are installed on a node, new property settings are added.

WebSphere Application Server system management uses the managed object metadata properties as follows:

- To display the node version in the administrative console
- To ensure that new configuration types or attributes are not created or set on older release nodes
- To ensure that new resource types are not created on old release nodes
- To ensure that new applications are not installed on old release nodes because the old run time cannot support the new applications

For detailed information about the following properties, see the Application Server application programming interface (API).

com.ibm.websphere.baseProductShortName

The product short name for the WebSphere Application Server that is installed.

com.ibm.websphere.baseProductVersion

The version of WebSphere Application Server that is installed.

com.ibm.websphere.nodeOperatingSystem

The operating system platform on which the node runs.

com.ibm.websphere.nodeSysplexName
The sysplex name on a z/OS operating system.

This property applies to the z/OS operating system only.

Starting and stopping a node

This topic describes how you can start and stop a node. Starting and stopping a node is applicable only if your profile, also known as a node, is added to a WebSphere Application Server Network Deployment domain or cell.

Before you can start and stop a node, you must federate the node into a cell.

Start or stop a node as need in administering your Network Deployment environment. Before your environment can service requests, you must have the deployment manager and node started, and typically an HTTP server.

- Start a node.

  Use one of these methods to start a node:

  – Use the startNode Qshell script.

    For information on this script, see the startNode command in the *Using the administrative clients* PDF.

  – Use the Submit Job (SBMJOB) CL command.

    You can run this CL command from an i5/OS command line:

    ```
    SBMJOB CMD(CALL PGM(QWAS61/QWASSTRSVR) PARM('-profilePath'
      'profile_root' '-server' 'nodeagent')) JOB(nodeagent)
      JOBD(QWAS61/QWASJOBD) JOBQ(QWAS61/QWASJOBQ) USER(QEJBSVR) LANGID(*USRPRF)
      CNTRYID(*USRPRF) CCSID(*USRPRF) OUTQ(QWAS61/QWASOUTQ)
    ```

    where profile_root is the profile root of the node agent.

- Stop a node.

  Use one of these methods to stop a node:

  – Use the stopNode Qshell script.

    You can use the stopNode script to stop a node from the Qshell command line of the iSeries server hosting the node. For more information about using this script, see the stopNode command in the *Using the administrative clients* PDF.

  – Use the deployment manager administrative console.

    To use the deployment manager administrative console to stop a node:

    1. Start the Network Deployment profile that manages your node.
    2. Start the administrative console for the deployment manager. For more information, see the Starting the administrative console for the deployment manager topic in the *Installing your application serving environment* PDF.
    3. Expand **System Administration** and click **Node Agents**.
    4. Select the checkbox for the node that you want to stop.
    5. Click **Stop**.

  – Use the End Job (ENDJOB) CL command

    To use the ENDJOB CL command to end an application server, enter this command on an i5/OS command line:

    ```
    ENDJOB JOB(jobNumber/QEJBSVR/jobName) OPTION(*CNTRLD) DELAY(delayTime)
    ```

    where *jobNumber* is the job number, *jobName* is the name of the application server job for the deployment manager, and *delayTime* is the amount of time to wait for the job to end in seconds. Set

the value to 600 seconds initially. To find out what the appropriate delayTime value is, see “Shutting down the WebSphere Application Server subsystem” on page 207.

You have started and stopped a node.

You can do a variety of things, such as deploy applications, create a cluster, and generally administer your Network Deployment environment.

Node group

A *node group* is a collection of managed nodes. Managed nodes are WebSphere Application Server nodes. A node group defines a boundary for server cluster formation.

Nodes that you organize into a node group need to be similar in terms of installed software, available resources, and configuration to enable servers on those nodes to host the same applications as part of a server cluster. The deployment manager does no validation to guarantee that nodes in a given node group have anything in common.

Node groups are optional and are established at the discretion of the WebSphere Application Server administrator. However, a node must be a member of a node group. Initially, all Application Server nodes are members of the default DefaultNodeGroup node group.

A node can be a member of more than one node group.

On the z/OS platform, an Application Server node must be a member of a sysplex node group. Nodes in the same sysplex must be in the same sysplex node group. A node can be in one sysplex node group only.

Nodes on distributed platforms and i5/OS platforms cannot be members of a node group that contains a node on a z/OS platform. However, nodes on distributed platforms and nodes on i5/OS platforms can be members of the same node group.

To delete a node group, the node group must be empty. The default node group cannot be deleted.

Node group membership rules

Nodes can be members of node groups if they meet certain requirements.

Node group membership must adhere to the following rules:

- A node in a node group must be a managed node.
- A managed node must be a member of at least one node group.
- Nodes on distributed platforms and nodes on i5/OS platforms can be members of the same node group.
- Nodes on distributed platforms and i5/OS platforms cannot be members of a node group that contains a node on a z/OS platform.

Examples: Using node groups

Use node groups to define groups of nodes that are capable of hosting members of the same cluster. An application that is deployed to a cluster must be capable of running on any of the cluster members. The node that hosts each of the cluster members must be configured with software and settings that are necessary to support the application.

By organizing nodes that satisfy your application requirements into a node group, you establish an administrative policy that governs which nodes can be used together to form a cluster. The people who define the cell configuration and the people who create server clusters can operate with greater independence from one another, if they are different people.

Example 1

Assume the following information:

- A cell is comprised of nodes one to eight.
- Each node is a managed node, which means that each node is configured with an Application Server.
- Nodes six, seven, and eight are additionally configured as WebSphere Business Integration Server Foundation nodes.
- All nodes are either z/OS system nodes from the same sysplex, or distributed platform nodes.

  An i5/OS node is considered a distributed platform node.
- By default, all the nodes are in the default DefaultNodeGroup node group.

Applications that exploit WebSphere Business Integration Server Foundation functions can run successfully only on nodes six, seven, and eight. Therefore, clusters that host these applications can be formed only on nodes six, seven, and eight. To define a clustering policy that guides users of your WebSphere cell into building clusters that can span only predetermined nodes, create an additional node group called WBINodeGroup, for example. Add to the node group nodes six, seven, and eight. If you create a cluster on a node from the WBINodeGroup node group, the system allows only nodes from the WBINodeGroup node group to be members of the cluster.

Example 2

i5/OS nodes are considered distributed platform nodes.

Assume the following information:

- A cell is comprised of nodes one to six.
- Each node is a managed node, which means that each node is configured with an Application Server.
- Nodes one to four are on distributed platforms.
- Nodes five and six are nodes on the z/OS operating system and are in the PLEX1 sysplex.
- The deployment manager is on a distributed platform node.
- Nodes one to four are members of the DefaultNodeGroup node group by default.
- You created empty PLEX1NodeGroup node group to group the z/OS operating system nodes on the PLEX1 sysplex.
- You joined the nodes on the z/OS operating system to the PLEX1NodeGroup node group when you added them to the cell. Nodes on the z/OS operating system cannot be in the same node group with the distributed platform nodes.

Applications that exploit z/OS functions in the PLEX1 sysplex can run successfully on nodes five and six only. Therefore, clusters that host these applications can be formed only on nodes five and six. The required separation of distributed platform nodes from z/OS system nodes establishes a natural clustering policy that guides users of your Application Server cell into building clusters that can span only predetermined nodes. If you create a cluster on a node from the PLEX1NodeGroup node group, the system allows only nodes from the PLEX1NodeGroup node group to be members of the cluster.

Managing node groups

This task discusses how to create and manage node groups.

Read about Nodes groups if you are unfamiliar with them.

Your WebSphere Application Server environment has a default node group. However, if you need additional node groups to manage your Application Server environment, you can create and configure additional node groups. You can delete a node group as long as it is not a default node group.

- View and configure node groups.
  1. Click **System Administration > Node groups** in the console navigation tree.
  2. To view additional information about a particular node group or to further configure a node group, click on the node group name under **Name**.
- Create a node group.
  1. Click **System Administration > Node groups** in the console navigation tree.
  2. Click **New**.
  3. Specify the node group name and description.

  The node group is added to the WebSphere Application Server environment . The name of the node group appears in the name column of the Node group page.

  You can now add nodes to the node group.
- Delete a node group if the node group is not the default node group.
  1. If the node group contains members, delete the members:
     a. Click **System Administration > Node groups** in the console navigation tree.
     b. Under **Name**, click the node group whose members you want to delete.
     c. Click **Node group members**.
     d. Select all the node group members.
     e. Click **Remove**.
  2. Click **System Administration > Node groups**.
  3. Select an empty node group.
  4. Click delete.

Node group collection

Use this page to manage node groups. A node group is a collection of WebSphere Application Server nodes. A node group defines a boundary for server cluster formation.

Nodes that are organized into a node group should be enough alike in terms of installed software, available resources, and configuration to enable servers on those nodes to host the same applications as part of a server cluster. The deployment manager does no validation to guarantee that nodes in a given node group have anything in common.

Node groups are optional and are established at the discretion of the WebSphere administrator. However, a node must be a member of a node group. Initially, all Application Server nodes are members of the default node group. The default node group is DefaultNodeGroup.

A node can be a member of more than one node group.

On the z/OS platform, an Application Server node must be a member of a sysplex node group. Nodes in the same sysplex must be in the same sysplex node group. A node can only be in one sysplex node group. Sysplex node groups are special node groups that the system manages.

A node on a distributed platform and a node on a z/OS platform cannot be members of the same node group.

To delete a node group, the node group must be empty. The default node group cannot be deleted.

To view this administrative console page, click **System Administration > Node groups**.

Name

Specifies a name for a node group that is unique within the cell.

Members
Specifies the number of members or nodes in the node group.

Description
Specifies a description that you define for the node group.

Node group settings
Use this page to view or change the configuration or topology settings for a node group instance.

To view this administrative console page, click **System Administration > Node groups >** *node group name*.

Name:

Specifies a logical name for the node group. The name must be unique within the cell. The name can start with a number.

Data type	String
Maximum length	64 characters

Short name:

Specifies the name of a node. The name must contain 1-8 characters, which are either alphanumeric or national language. It cannot start with a number.

On the z/OS system the short name property is:
- Read-only
- Used only by sysplex node groups
- Defined during installation and customization

Sysplex:

Specifies the name of a node. The name is eight characters, alphanumeric or national language. It cannot start with a numeric. It is used only by sysplex node groups on the z/OS platform. It is defined during installation and customization on z/OS platforms only.

The Sysplex property is read only.

Members:

Specifies the number of nodes within the node group.

Data type	Integer

Description:

Specifies the description that you define for the node group. The description has no specific maximum length.

Managing node group members

Use this topic to manage the nodes in your node groups by viewing, adding or deleting the nodes in a node group.

Read about Nodes groups and Node group membership rules if you are unfamiliar with them.

All nodes must be a member of at least one node group. Initially, all Application Server nodes are members of the default node group named DefaultNodeGroup. Make the nodes that you organize into a node group enough alike in terms of installed software, available resources, and configuration to enable servers on those nodes to host the same applications as part of a server cluster.

- View node groups members.
  1. Click **System Administration > Node groups > *node group name* > Nodes > Node group members** in the console navigation tree.
  2. To view additional information about a particular node group member for this node group, click on the node group member name under **Name**.
- Add a node to a node group.
  1. Click **System Administration > Node groups > *node group name* > Nodes > Node group members** in the console navigation tree.
  2. Click **Add**.
  3. Select the node from a list. The node group member name is the node name.

  The node group member is added to the node group specified on the breadcrumb trail. The name of the node group member appears in the name column of the Node group member page. You can add additional nodes of similar characteristics to the node group by repeating the steps for adding a node to a node group.

  If the node you add does not satisfy the node group membership rules for the target node group, the add node operation fails with an error message.
- Remove a node from a node group.
  1. Click **System Administration > Node groups > *node group name* > Nodes > Node group members** in the console navigation tree.
  2. Select the box next to each node group member that you want to remove from the node group.
  3. Click **Remove**.

  Each node group member that you selected is removed from the node group specified on the breadcrumb trail.

Node group member collection

Use this page to manage node groups members. A node group member is a WebSphere Application Server node.

Click **Add** to add node members to the node group. Click **Remove** to remove node members from the node group.

To view this administrative console page, click **System Administration > Node groups >** *node group name* **> Node group members**.

Name

Specifies the name of a node group member.

Node group member settings

Use this page to view or change the configuration or topology settings for a node group member.

To view this administrative console page, click **System Administration > Node groups >** *node group name* **> Node group members >** *node group member name*.

Name:

Specifies a logical name for the node group member. A node group member is a node. The name must be unique within the cell.

A node group member name usually is identical to the host name for the computer.

Data type	String
Maximum length	64 characters

The name must contain alphanumeric or national language characters and can start with a number.

Node agents

Node agents are administrative agents that route administrative requests to servers.

A node agent is a server that runs on every host computer system that participates in the WebSphere Application Server Network Deployment product. It is purely an administrative agent and is not involved in application serving functions. A node agent also hosts other important administrative functions such as file transfer services, configuration synchronization, and performance monitoring.

Managing node agents

Node agents are administrative agents that represent a node to your system and manage the servers on that node. Node agents monitor application servers on a host system and route administrative requests to servers. A node agent is created automatically when a node is added to a cell. This topic describes how to view information about a node agent, stop and start the processing of a node agent, and stop and restart application servers on the node that is managed by the node agent.

Before you can manage a node agent, you must install the Network Deployment product.

You can manage nodes through the wsadmin scripting tool, through the Java application programming interfaces (APIs), or through the administrative console. Perform the following tasks to manage nodes on an application server through the administrative console.

- View information about a node agent. Use the Node Agents page. Click **System Administration > Node Agents** in the console navigation tree. To view additional information about a particular node agent or to further configure a node agent, click the node agent name under **Name**.

  IP verions: WebSphere Application Server on other platforms has support for Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6). However, the i5/OS platform supports only IPv4. When you add a node to a cell, the format in which you specify the host name is based on the version of IP the node will be using. See “IP version considerations for cells” on page 134.
- Stop and then restart the processing of a node agent. On the Node Agents page, select the check box beside the node agent that you want to restart; then click **Restart**. It is important to keep a node agent running because a node agent must be running for application servers on the node managed by the node agent to run.
- Stop and then restart all application servers on the node that is managed by the node agent. On the Node Agents page, select the check box beside the node agent that manages the node whose servers you want to restart, then click **Restart all Servers on Node**.

  Note that the node agent for the node must be processing to restart application servers on the node.
- Stop the processing of a node agent. On the Node Agents page, select the check box beside the node agent that you want to stop processing; then click **Stop**.

Depending on the steps that you completed, you have viewed information about a node agent, stopped and started the processing of a node agent, and stopped and restarted application servers on the node that is managed by the node agent.

You can administer other aspects of the Network Deployment environment, such as the deployment manager, nodes, and cells.

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.

To view this administrative console page, click **System Administration > Node Agents** .

Name

Specifies a logical name for the node agent server.

Node

Specifies a name for the node. The node name is unique within the cell.

A node name usually is identical to the host name for the computer. That is, a node usually corresponds to a physical computer system with a distinct IP host address.

However, the node name is a purely logical name for a group of servers. You can name the node anything you please. The node name does not have to be the host name.

Both Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) are now supported by WebSphere Application Server, but there are restrictions that apply to using both IPv4 and IPv6 in the same cell. Note that when a node is added to a cell, the format in which the name is specified is based on the version of IP the node will be using.

Version

Specifies the product version of the node.

The product version is the version of a WebSphere Application Server node agent and Application Servers that run on the node.

Status

Indicates whether the node agent server is started or stopped.

Note that if the status of servers such application servers is *Unavailable*, the node agent is not running in the servers' node and you must restart the node agent before you can start the servers.

Node agent server settings

Use this page to view information about and configure a node agent. A node agent coordinates administrative requests and event notifications among servers on a machine. A node agent is the running server that represents a node in a Network Deployment environment.

To view this administrative console page, click **System Administraton > Node Agents >** *node_agent_name*.

A node agent must be started on each node in order for the deployment manager node to be able to collect and control servers configured on that node. If you use configuration synchronization support, a node agent coordinates with the deployment manager server to synchronize the node's configuration data with the master copy managed by the deployment manager.

You must initially start a node agent outside the administrative console. For information on how to initially start a node agent, see the WebSphere Application Server Information Center.

The Runtime tab displays only when a node agent runs.

Name:

Specifies a logical name for the node agent server.

Data type String

Node Name:

Specifies the name of the node for the node agent server.

Both Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) are now supported by WebSphere Application Server, but there are restrictions that apply to using both IPv4 and IPv6 in the same cell. Note that when a node is added to a cell, the format in which the name is specified is based on the version of IP the node will be using.

Data type String

Short name:

Specifies the short name of the node agent 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 node agent 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 node agent server.

Data type String
Default *host_name*Network

Node Name:

Specifies the name of the node for the node agent server.

Both Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) are now supported by WebSphere Application Server, but there are restrictions that apply to using both IPv4 and IPv6 in the same cell. Note that when a node is added to a cell, the format in which the name is specified is based on the version of IP the node will be using.

Data type String

State:

Indicates whether the node agent server is started or stopped.

Data type String
Default Started

Administration service settings

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

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

Preferred Connector

Specifies the preferred JMX Connector type. Available options, such as SOAPConnector or RMIConnector, are defined using the JMX Connectors page.

Data type String
Default SOAP

Extension MBean Providers collection

Use this page to view and change the configuration for JMX extension MBean providers.

You can configure JMX extension MBean providers to be used to extend the existing WebSphere managed resources in the core administrative system. Each MBean provider is a library containing an implementation of a JMX MBean and its MBean XML Descriptor file.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Administration > Administration Services > Extension MBean Providers**

Name The name used to identify the Extension MBean provider library.

Description
An arbitrary descriptive text for the Extension MBean Provider configuration.

Classpath
The path to the Java archive (JAR) file that contains the Extension MBean provider library. This class path is automatically added to the Application Server class path.

Extension MBean Provider settings

Use this page to view and change the configuration for a JMX extension MBean provider.

You can configure a library containing an implementation of a JMX MBean, and its MBean XML Descriptor file, to be used to extend the existing WebSphere managed resources in the core administrative system

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Administration > Administration Services > Extension MBean Providers >** *provider_library_name*

Classpath:

The path to the Java archive (JAR) file that contains the Extension MBean provider library. This class path is automatically added to the Application Server class path. The class loader needs this information to load and parse the Extension MBean XML Descriptor file.

Data type String

Description:

An arbitrary descriptive text for the Extension MBean Provider configuration. Use this field for any text that helps identify or differentiate the provider configuration.

Data type String

Name:

The name used to identify the Extension MBean provider library.

Data type String

Extension MBean collection

You can configure Java Management Extension (JMX) MBeans to extend the existing WebSphere Application Server managed resources in the administrative console. Use this page to register JMX MBeans. Any MBeans that are listed have already been registered.

To view this administrative console page, click **Servers > Application Servers >** *server name* **> Administration > Administration Services > Extension MBean Providers >** *provider library name***> extensionMBeans**

DescriptorURI
Specifies the location, relative to the provider class path, where the MBean XML descriptor file is located.

Type Specifies the type to use for registering this MBean. The type must match the type that is declared in the MBean descriptor file.

Extension MBean settings

Use this page to view and configure Java Management Extension (JMX) MBeans.

To view this administrative console page, click **Servers > Application Servers >** *server name* **> Administration > Administration Services > Extension MBean Providers >** *provider library name* **> ExtensionMBeans >** *descriptorURI*

descriptorURI:

Specifies the location, relative to the provider class path, where the MBean XML descriptor file is located.

Data type String

type:

Specifies the type to use for registering this MBean. The type must match the type that is declared in the MBean descriptor file.

Data type String

Java Management Extensions connector properties

You can specify or set a property in the administrative console, the wsadmin tool, Application Server commands, the scripts that run from a command-line interface, or a custom Java administrative client program that you write. You can also set SOAP connector properties in the `soap.client.props` file.

A Java Management Extensions (JMX) connector can either be a Remote Method Invocation (RMI) connector or a Simple Object Access Protocol (SOAP) connector.

For specific information on how to code the JMX connector properties for the wsadmin tool, the Application Server commands, or scripts, see the particular tool or command. For specific information on how to code the JMX connector properties for a custom Java administrative client program, see the Java API documentation for Application Server.

For the administrative console, this topic specifies the coding of the particular setting or property. Coding of properties in the `soap.client.props` file that are specific to JMX connectors is specified. These properties begin with com.ibm.SOAP. Other properties in the `soap.client.props` file that contain information that can be set elsewhere in the Application Server are not documented here. The coding for the com.ibm.ssl.contextProvider property, which can be set only in the `soap.client.props` file, is specified.

Each profile has a property file at *profile_root*`/properties/soap.client.props`, where *profile_root* is the fully qualified path of the directory that contains your profile. These property files allow you to set different properties, including security and timeout properties. These properties are the default for all the administrative connections that use the SOAP JMX connector between processes running in a particular profile. For instance, the wsadmin program that runs under a particular profile uses the property values from that file for the SOAP connector behavior unless the properties are overridden by some other programmatic means.

To view the JMX connector custom properties administrative console panel that goes with this article, click **Servers > Application servers >***server name* **> Server Infrastructure > Administration > Administration Services > Additional properties > JMX Connectors>***connector type* **> Additional Properties > Custom properties**.

SOAP connector properties

This section discusses JMX connector properties that pertain to SOAP connectors.

SOAP request timeout

The value that you choose depends on a number of factors, such as the size and the number of the applications that are installed on the server, the speed of your machine, and the usage of your machine.

The program default value for the request timeout is `600 seconds`. However, other components that connect to the SOAP client can override the default. Components that use the `soap.client.props` file have a default value of `180 seconds`.

Set the property by using one of the following options:

- Scripts that run from a command-line interface.

- The `soap.client.props` file.

Property	com.ibm.SOAP.requestTimeout
Data type	Integer
Range in seconds	0 to n If the property is zero (0), the request never times out.
Default	180

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	requestTimeout
Data type	Integer
Range in seconds	0 to n If the property is zero (0), the request never times out.
Default	600

- A Java administrative client. The property is AdminClient.CONNECTOR_SOAP_REQUEST_TIMEOUT.

Configuration URL

Specify the configuration Universal Resource Locator (URL) property if you want a program to read SOAP properties from this file. You can set the property by using one of the following options:

- Scripts run from a command-line interface. Scripts can pass the Configuration URL property to the Application Server on the com.ibm.SOAP.ConfigURL system property.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	ConfigURL
Data type	String
Valid Value	http://*Path*/soap.client.props
Default	None

- A Java administrative client. Use the AdminClient.CONNECTOR_SOAP_CONFIG property.

Security context provider

This property indicates the Secure Sockets Layer (SSL) implementation to use between the Application Server and the SOAP client. You can specify either IBM Java Secure Sockets Extension (IBMJSSE) or IBM Java Secure Sockets Extension that has undergone Federal Information Processing Standards certification (IBMJSSEFIPS). For information about IBMJSSEFIPS, see the *Using the administrative clients* PDF.

Set the property by using the `soap.client.props` file.

Property	com.ibm.ssl.contextProvider
Data type	String
Valid Values	`IBMJSSE` `IBMJSSEFIPS` `IBMJSSE2`
Default	IBMJSSE2

Secure Sockets Layer (SSL) security

Use this property to enable SSL security between Application Server and the SOAP client. Set the property by using one of the following options:

- Scripts that run from a command-`line interface.
- The `soap.client.props` file.

Property	com.ibm.SOAP.securityEnabled
Data type	Boolean
Default	False

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	securityEnabled
Data type	Boolean
Default	False

- A Java administrative client. Use the AdminClient.CONNECTOR_SECURITY_ENABLED property.

SOAP and RMI connector properties

This section discusses JMX connector properties that pertain to both SOAP connectors and RMI connectors.

Connector type

A connector type of SOAP or RMI, depends on whether Application Server connects to a SOAP server or an RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts that run from a command-line interface.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	Type
Data type	String
Valid values	`SOAPConnector` `RMIConnector`
Default	SOAPConnector

- A Java administrative client. Use the AdminClient.CONNECTOR_TYPE property. Specify the connector type by using the AdminClient.CONNECTOR_TYPE_RMI or the AdminClient.CONNECTOR_TYPE_SOAP constants.

Host

The host name or the IP address of the server to which Application Server connects. The server can be a SOAP server or an RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts that run from a command-line interface.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	host
Data type	String
Valid values	Host name or IP address
Default	None

- A Java administrative client. Use the AdminClient.CONNECTOR_HOST property.

Port

The port number of the server to which Application Server connects. The server can be a SOAP server or an RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command-line interface.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	port
Data type	Integer
Valid value	Port number
Default	None

- A Java administrative client. Use the AdminClient.CONNECTOR_PORT property.

User name

The user name that Application Server uses to access the SOAP server or the RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command-line interface.
- The `soap.client.props` file.

Property	com.ibm.SOAP.loginUserid
Data type	String
Valid value	The value must match the global SSL settings for SOAP or RMI.
Default	None

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	username
Data type	String
Valid value	The value must match the global SSL settings for SOAP or RMI.
Default	None

- A Java administrative client. Use the AdminClient.USERNAME property.

Password

The password that Application Server uses to access the SOAP server or the RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command-line interface.
- The `soap.client.props` file.

Property	com.ibm.SOAP.loginPassword
Data type	String

Valid values	The value must match the global SSL settings for SOAP or RMI.
Default	None

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	password
Data type	String
Valid values	The value must match the global SSL settings for SOAP or RMI.
Default	None

- A Java administrative client. Use the AdminClient.PASSWORD property.

RMI connector properties

This section discusses JMX connector properties that pertain to RMI connectors.

Disabling the JSR 160 RMI connector

Support for JMX Remote application programming interface (JSR 160) is enabled by default so that you automatically receive specification-compliant JMX function. To disable the function for a particular server, set the property by using one of the following options:

- The wsadmin tool.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	disableJDKJMXConnector
Data type	string
Value	true

Java Management Extensions connectors

Use this page to view and change the configuration for Java Management Extensions (JMX) connectors.

To view this administrative console page, click one of the following paths:

- **Servers > Application Servers >** *server_name* **> Administration > Administration Services > JMX Connectors**
- **Servers > JMS Servers >** *server_name* **> Administration > Administration Services > JMX Connectors**

Java Management Extensions (JMX) connectors communicate with WebSphere Application Server when you invoke a scripting process. There is no default for the type and parameters of a connector. The `wsadmin.properties` file specifies the Simple Object Access Protocol (SOAP) connector and an appropriate port number. You can also use the Remote Method Invocation (RMI) connector.

Use one of the following methods to select the connector type and attributes:

- Specify properties in a properties file.
- Indicate options on the command line.

Type

Specifies the type of the JMX connector.

Data type	Enum

Default SOAPConnector

Range

SOAPConnector
For JMX connections using Simple Object Access Protocol (SOAP).

RMIConnector
For JMX connections using Remote Method Invocation (RMI).

JMX connector settings

Use this page to view the configuration for a Java Management Extensions (JMX) connector.

To view this administrative console page, click one of the following paths:

- **Servers > Application Servers >** *server_name* **> Administration > Administration Services > JMX Connectors >** *connector_type*
- **Servers > JMS Servers >** *server_name* **> Administration > Administration Services > JMX Connectors >** *connector_type*

Type:

Specifies the type of the JMX connector.

Data type Enum

Default SOAPConnector

Range

SOAPConnector
For JMX connections using Simple Object Access Protocol (SOAP).

RMIConnector
For JMX connections using Remote Method Invocation (RMI).

Repository service settings

Use this page to view and change the configuration for an administrative service repository.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **Administration > Administration Services > Repository Service**.

Audit Enabled

Specifies whether to audit repository updates in the log file. The default is to audit repository updates.

Data type Boolean

Default true

Administration services custom properties

This topic discusses the administration services custom properties that you can set on the administrative console.

To view the administration services custom properties administrative console page that goes with this topic, click: **Servers > Application Server >** *server_name* **> Administration > Administration Services > Custom Property**.

Specify a property and its value as a name-value pair on the Administration services custom properties page.

Disable routing

When a custom managed bean (MBean) is registered directly with the MBean server that runs in a WebSphere Application Server process, the MBean object name is enhanced by default to include the cell, node, and process names as key properties.

With this enhancement, in a Network Deployment environment, the MBean that is registered on an application server is addressable through a client that is connected to the deployment manager.

To turn off the default behavior, set the following custom property on the application server:

Property name
com.ibm.websphere.mbeans.disableRouting

Data type
string

Value
One or more MBean object names tagged with `<on>...</on>`. You can specify the object name of your MBean or a pattern that matches the names of several MBeans.

Example:

If you register a custom MBean with the `WebSphere:type=custom,name=custommbean1` object name and another custom MBean with the `WebSphere:type=custom,name=custommbean2` object name, each of the following values is valid:

- `<on>WebSphere:type=custom,name=custommbean1</on>`

  The value disables the MBean object name modification for this MBean.
- `<on>WebSphere:type=custom,*</on>`

  The value disables the MBean object name modification for this MBean.
- `<on>WebSphere:type=custom,name=custommbean1</on><on>WebSphere:type=custom,name=custommbean2</on>`

  The value disables the object name modification for both MBeans.

If this custom property is set, an administrative client needs to connect directly to the application server on which the MBean is registered to invoke methods. The MBean cannot participate in all the distributed functions of the administrative system.

Administrative audits

This topic discusses aspects of administrative audits, such as log files that contain the audit information, the administrative actions that are audited, and the types of audit messages that are logged.

Administrative audits use the same logging facility as the rest of the product. The audits are available in both the `activity.log` file and the `SystemOut.log` of the server that performs the action. You do not need to enable trace to produce the audits. However, through the Repository service console page, you can control whether configuration change auditing is done. This type of audit is done by default. Operational command auditing is always enabled. Information about which user performed the change is available only when security is enabled.

The following administrative actions are audited:

- All configuration changes, in terms of the configuration documents that are created, modified, or deleted.

- Certain operational changes like starting and stopping nodes, clusters, servers, and applications. These managed bean (MBean) operations provide administrative auditing:

Table 13.

MBean type	MBean operations
CellSync	syncNode
Cluster	start, stop, stopImmediate, rippleStart
NodeAgent	launchProcess, stopNode, restart
Server	stop, stopImmediate
AppManagement	startApplication, stopApplication

Configuration change audits have ADMRxxxxI message IDs, where xxxx is the message number. Operational audits have ADMN10xxI message IDs, where 10xx is the message number.

Here are some examples from the deployment manager `SystemOut.log` file:

```
[7/23/03 17:04:49:089 CDT] 39c26dad FileRepositor A ADMR0015I: Document
cells/ellingtonNetwork/security.xml was modified by user u1.
   [7/23/03 17:04:49:269 CDT] 3ea0edb5 FileRepositor A ADMR0016I: Document
cells/ellingtonNetwork/nodes/ellington/app.policy was created by user u1.
   ...
   [7/23/03 17:13:54:081 CDT] 39a572a1 AdminHelper   A ADMN1008I: Attempt
made to start the SamplesGallery application. (User ID = u1)
   ...

and from the node agent SystemOut.log...
[7/23/03 17:38:43:461 CDT]  23d1326 AdminHelper   A ADMN1000I: Attempt
made to launch server1 on node ellington. (User ID = u1)

and from the app server SystemOut.log...
[7/23/03 17:39:59:360 CDT] 24865373 AdminHelper   A ADMN1020I: Attempt
made to stop the server1 server. (User ID = u1)
```

The message text is split for printing purposes.

Remote file services

Configuration documents describe the available application servers, their configurations, and their contents. Two file services manage configuration documents: the file transfer service and the file synchronization service.

The following information describes what the file services do:

File transfer service

The file transfer service enables the moving of files between the network manager and the nodes. It uses the HTTP protocol to transfer files. When you enable security in the WebSphere Application Server product, the file transfer service uses certificate-based mutual authentication. You can use the default key files in a test environment. Ensure that you change the default key file to secure your system.

The ports used for file transfer are the HTTP_Transport port, the HTTPS transport port, the administrative console port, and the administrative console secure port. For more information, see “Port number settings in WebSphere Application Server versions” on page 1.

File synchronization service

The file synchronization service ensures that a file set on each node matches that on the deployment manager node. This service promotes consistent configuration data across a cell. You can adjust several configuration settings to control file synchronization on individual nodes and throughout a system.

This service runs in the deployment manager and node agents, and ensures that configuration changes made to the cell repository are propagated to the appropriate node repositories. The cell repository is the master repository, and configuration changes made to node repositories are not propagated up to the cell. During a synchronization operation a node agent checks with the deployment manager to see if any configuration documents that apply to the node have been updated. New or updated documents are copied to the node repository, and deleted documents are removed from the node repository.

The default behavior, which is enabled, is for each node agent to periodically run a synchronization operation. You can configure the interval between operations or disable the periodic behavior. You can also configure the synchronization service to synchronize a node repository before starting a server on the node.

Configuring remote file services

Configuration data for the WebSphere Application Server product resides in files. Two services help you reconfigure and otherwise manage these files: the file transfer service and file synchronization service.

By default, the file transfer service is always configured and enabled at a node agent, so you do not need to take additional steps to configure this service. However, you might need to configure the file synchronization service.

1. Go to the File Synchronization Service page. Click **System Administration > Node Agents** in the console navigation tree. Then, click the node agent for which you want to configure a synchronization server and click **File Synchronization Service**.
2. On the File Synchronization Service page, customize the service that helps make configuration data consistent across a cell by moving updated configuration files from the deployment manager to the node. Change the values for properties on the File Synchronization Service page. The file synchronization service is always started, but you can control how it runs by changing the values.

File transfer service settings

Use this page to configure the service that transfers files from the deployment manager to individual remote nodes.

To view this administrative console page, click **System Administration > Node Agents >** *node_agent_name* **> File Transfer Service**.

Enable service at server startup

Specifies whether the server attempts to start the specified service. Some services are always enabled and disregard this property if set. This setting is enabled by default.

Data type Boolean
Default true

Retries count

Specifies the number of times you want the file transfer service to retry sending or receiving a file after a communication failure occurs.

Data type Integer
Default 3

If the retries count setting is blank, the file transfer service sets the default to 3. If the retries count setting is `0`, the file transfer service does not retry. The default is the recommended value.

Retry wait time
Specifies the number of seconds that the file transfer service waits before it retries a failed file transfer.

Data type Integer
Default 10

If the retry wait time setting is blank, the code sets the default to 10. If the retry wait time setting is `0`, the file transfer service does not wait between retries. The default is the recommended value.

File synchronization service settings
Use this page to specify that a file set on one node matches that on the central deployment manager node and to ensure consistent configuration data across a cell.

You can synchronize files on individual nodes or throughout your system.

To view this administrative console page, click **System Administration > Node Agents >** *node_agent_name* **> File Synchronization Service**.

Enable service at server startup
Specifies whether the server attempts to start the file synchronization service. This setting does not cause a file synchronization operation to start. This setting is enabled by default.

Data type Boolean
Default true

Synchronization Interval
Specifies the number of minutes that elapse between synchronizations. Increase the time interval to synchronize files less often. Decrease the time interval to synchronize files more often.

Data type Integer
Units Minutes
Default 1
The minimum value that the application server uses is 1. If you specify a value of 0, the application server ignores the value and uses the default of 1.

Automatic Synchronization
Specifies whether to synchronize files automatically after a designated interval. When this setting is enabled, the node agent automatically contacts the deployment manager every synchronization interval to attempt to synchronize the node's configuration repository with the master repository owned by the deployment manager.

If the Automatic synchronization setting is enabled, the node agent attempts file synchronization when it establishes contact with the deployment manager. The node agent waits the synchronization interval before it attempts the next synchronization.

Remove the check mark from the check box if you want to control when files are sent to the node.

Data type Boolean
Default true

Startup Synchronization
Specifies whether the node agent attempts to synchronize the node configuration with the latest configurations in the master repository prior to starting an application server.

The default is to not synchronize files prior to starting an application server. Enabling the setting ensures that the node agent has the latest configuration but increases the amount of time it takes to start the application server.

Note that this setting has no effect on the startServer command. The startServer command launches a server directly and does not use the node agent.

Data type	Boolean
Default	false

Exclusions
Specifies files or patterns that should not be part of the synchronization of configuration data. Files in this list are not copied from the master configuration repository to the node, and are not deleted from the repository at the node.

The default is to have no files specified.

iSeries users: The default is `*/plugin-cfg.xml`, which excludes the Web Server plug-in configuration file from synchronization.

To specify a file, use a complete name or a name with a leading or trailing asterisk (*) for a wildcard. For example:

`cells/`*`cell name`*`/nodes/`*`node name`*`/`*`file name`*	Excludes this specific file
`*/`*`file name`*	Excludes files named *file name* in any context
`dirname/*`	Excludes the subtree under `dirname`

Press **Enter** at the end of each entry. Each file name appears on a separate line.

Since these strings represent logical document locations and not actual file paths, only forward slashes are needed no matter the platform.

Changes to the exclusion list are picked up when the node agent is restarted.

Data type	String
Units	File names or patterns

Administrative agents: Resources for learning

Use the following links to find relevant supplemental information about WebSphere Application Server administrative agents and distributed administration. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

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

View links to additional information:

Administration

- IBM WebSphere Application Server Redbooks

  This site contains a listing of all WebSphere Application Server Redbooks.
- IBM WebSphere developerWorks

  This site is the home of technical information for developers working with WebSphere products. You can download WebSphere software, take a fast path to developerWorks zones, such as VisualAge Java or WebSphere Application Server, learn about WebSphere products through a newcomers page, tutorials, technology previews, training, and Redbooks, get answers to questions about WebSphere products, and join the WebSphere community, where you can keep up with the latest developments and technical papers.
- WebSphere Application Server Support page

  Take advantage of the Web-based Support and Service resources from IBM to quickly find answers to your technical questions. You can easily access this extensive Web-based support through the IBM Software Support portal at URL `http://www-3.ibm.com/software/support/` and search by product category, or by product name. For example, if you are experiencing problems specific to WebSphere Application Server, click **WebSphere Application Server** in the product list. The WebSphere Application Server Support page appears.

Chapter 5. Configuring the environment

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

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

1. Configure virtual hosts.
2. Configure variables.
3. If your deployed applications use shared library files, define the shared library files needed.
   See “Managing shared libraries” on page 184.

Virtual hosts

When you configure WebSphere Application Server, you can associate a virtual host to one or more Web modules. Each Web module can be associated with one and only one virtual host.

A virtual host is a configuration entity that enables a single host machine to resemble multiple host machines. It maintains a list of Multipurpose Internet Mail Extensions (MIME) types that it processes. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine.

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

A client request for a servlet, JavaServer Pages file, or related resource contains a DNS alias and a Uniform Resource Indicator (URI) that is unique to that resource. When a client request for a servlet, JavaServer Pages file, or related resource is received, the DNS alias is compared to the list of all known virtual host groups to locate the correct virtual host, and the URI is compared to the list of all known URI groups to locate the correct URI group. If the virtual host group and URI group are found, the request is sent to the corresponding server group for processing and a response is returned to browser. If a matching virtual host group or URI group is not found, an error is returned to the browser.

The first time that you start an application server, a default virtual host (named default_host) is configured. The DNS aliases for the default virtual host are configured as `*:80` and `*:9080`, where port 80 is the HTTP server port and port 9080 is the port for the default server's HTTP transport. The default virtual host includes common aliases, such as the machine's IP address, short host name, and fully qualified host name. One of these aliases comprises the first part of the path for accessing a resource such as a servlet. For example, the alias `localhost:80` is used in the request `http://localhost:80/myServlet`.

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

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

You can use the administrative console to add or change DNS aliases if you want to use ports other than the default ports. If you do make a change to a DNS alias, you must regenerate the Web server plug-in configuration. You can use the administrative console to initiate the plug-in regeneration.

Note: You might want to add additional aliases or change the default aliases if:

- The HTTP server instance is running on a port other than 80. Add the correct port number to each of the aliases. For example, change `yourhost` to `yourhost:8000`.
- You want to make HTTPS requests, which use Secure Sockets Layer (SSL). To make HTTPS requests you must add port 443 to each of the aliases. Port 443 is the default port for SSL requests.
- Your Web server instance is listening for SSL requests on a port other than 443. In this situation, you must add that port number to each of the aliases.
- You want to use a port other then default port (9080) for the application server.
- You want to use other aliases that are not listed.

Why you would use virtual hosting

Virtual hosts let you manage a single application server on a single machine as if the application server were multiple application servers each on their own host machine. Resources associated with one virtual host cannot share data with resources associated with another virtual host. This is true even though the virtual hosts share the same application server on the same physical machine.

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

Suppose an Internet service provider (ISP) has two customers with Internet sites hosted on the same machine. The ISP keeps the two sites isolated from one another, despite their sharing a machine, by using virtual hosts. The ISP associates the resources of the first company with VirtualHost1 and the resources of the second company with VirtualHost2. Both virtual hosts map to the same application server.

Further suppose that both company sites offer the same servlet. Each site has its own instance of the servlet, and is unaware of the same servlet on the other site. If the company whose site is organized on VirtualHost2 is past due in paying its account with the ISP, the ISP can refuse all servlet requests that are routed to VirtualHost2. Even though the same servlet is available on VirtualHost1, the requests directed at VirtualHost2 do not go to the other virtual host.

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

You associate a servlet or other application with a virtual host instead of the actual DNS address.

The default virtual host (default_host)

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

The virtual host configuration uses wildcard entries with the ports for its virtual host entries.
- The default alias is *:80, using an internal port that is not secure.
- Aliases of the form *:9080 use the secure internal port.
- Aliases of the form *:9443 use the external port that is not secure.
- Aliases of the form *:443 use the secure external port.

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

How requests map to virtual host aliases

Virtual hosts let you manage a single application server on a single machine as if the application server were multiple application servers that are each on their own host machine. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even though the virtual hosts share the same application server on the same physical machine.

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

Mappings are both case sensitive and insensitive. For example, the portion `http://host:port/` is not case sensitive, but the URL that follows is case sensitive. The match must be alphanumerically exact. Also, different port numbers are treated as different aliases.

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

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

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

Two sets of associations occur for virtual hosts. Application deployment associates an application with a virtual host. Virtual host definitions associate the network address of the machine and the HTTP transport or Web server port assignment of the application server with the virtual host. Looking at the flow from the Web client request for the snoop servlet, for example, the following actions occur:

1. The Web client asks for the snoop servlet: at Web address `http://www.some_host.some_company.com:9080/snoop`
2. The some_host machine has the 9080 port assigned to the standalone application server, server1.
3. server1 looks at the virtual host assignments to determine the virtual host that is assigned to the alias `some_host.some_company.com:9080`.
4. The application server finds that no explicit alias for that DNS string exists. However, a wild card assignment for host name * at port 9080 does exist. This is a match. The virtual host that defines the match is default_host.
5. The application server looks at the applications deployed on the default_host and finds the snoop servlet.
6. The application server serves the application to the Web client and the requester is able to use the snoop servlet.

You can have any number of aliases for a virtual host. You can even have overlapping aliases, such as:

Virtual host	**Alias**	**Port**
default_host	*	9080
	localhost	9080
	my_machine	9080
	my_machine.my_company.com	9080
	localhost	80

The Application Server looks for a match using the explicit address specified on the Web client address. However, it might resolve the match to any other alias that matches the pattern before matching the explicit address. Simply defining an alias first in the list of aliases does not guarantee the search order when WebSphere Application Server is looking for a matching alias.

A problem can occur if you use the same alias for two different virtual hosts. For example, assume that you installed the default application and the snoop servlet on the default_host. You also have another virtual host called the admin_host. However, you have not installed the default application or the snoop servlet on the admin_host.

Assume that you define overlapping aliases for both virtual hosts because you accidentally defined port 9080 for the admin_host instead of port 9060:

Virtual host	Alias	Port
default_host	*	9080
	localhost	9080
admin_host	*	9060
	my_machine.com	9080

Assume that a Web client request comes in for `http://my_machine.com:9080/snoop`.

If the application server matches the request against `*:9080`, the application is served from the default_host. If the application server matches the request to `my.machine.com:9080`, the application cannot be found. A 404 error occurs in the browser that issues the request. A message states that the virtual host could not be found.

This problem is the result of not finding the requested application in the first virtual host that has a matching alias. The correct way to code aliases is for the alias name on an incoming request to match only one virtual host in all of your virtual host definitions. If the URL can match more than one virtual host, you can see the problem just described.

Configuring virtual hosts

For configuration purposes, a virtual host enables WebSphere Application Server to treat multiple host machines or port numbers as a single logical host (virtual host). You can combine multiple host machines into a single virtual host or assign host machines to different virtual hosts, to separate and control which WebSphere Application Server resources are available for client requests.

If your external HTTP server configuration uses the default port, 9080, you do not have to perform these steps.

You must update the HTTP port numbers associated with the default virtual host. or define a new virtual host and associate it with the ports your HTTP server configuration uses if:

- Your external HTTP server configuration uses a port other than the default port of 9080, you must define the port that you are using.
- You are using the default HTTP port 9080, but the port is no longer defined. You must define port 9080.
- You have created multiple application servers as either stand-alone servers or cluster members, and these servers use the same virtual host. Because each server must be listening on a different port, you must define a virtual host alias for the HTTP port of each server.

If you define new virtual host aliases, identify the port values that the aliases use on the “Host alias settings” on page 177 page in the administrative console.

To create a new virtual host or change the configuration of an existing virtual host:

1. In the administrative console, click **Environment > Virtual Hosts**.
2. **Optional:** Create a new virtual host. If you create a new virtual host, a default set of 90 MIME entries are automatically created for that virtual host.
   a. In the administrative console, click **New**.

b. Enter the name of the new virtual host and click **OK**. The new virtual host appears in the list of virtual hosts you can configure.

3. Select the virtual host whose configuration you want to change.
4. Under Additional Properties, click **Host Aliases**.
5. Create new host aliases or update existing host aliases to associate each of your HTTP port numbers with this virtual host.

   There must be a virtual host alias corresponding to each port your HTTP server configuration uses. There is one HTTP port associated with each Web container, and it is usually assigned to the virtual host named `default_host`. You can change the default assignment to any valid virtual host.

   The host aliases associated with the `default_host` virtual host are set to * when you install WebSphere Application Server. The * (an asterisk) indicates that the alias name does not have to be specified or that any name can be specified.

   When the URL for the application is entered into a Web browser, the port number is included. For example, if 9082 is the port number, the specified URL might look like the following:

   ```
   http://localhost:9082/wlm/SimpleServlet
   ```

   To create a new host alias:

   a. Click **New**.
   b. Specify a host alias name in the Host Name field and one of your HTTP ports in the Port field.

      You can specify * (an asterisk) for the alias name if you do not want to require the specification of the alias name or if you want to allow any name to be specified.
   c. Click **OK** and **Save** to save your configuration change.

   To update an existing host alias:

   a. Select an existing host alias name.
   b. Change the value specified in the Port field to one of your HTTP ports.
   c. Click **OK** and **Save** to save your configuration change.
6. **Optional:** Define a MIME object type and its file name extension if you require a MIME type other than the pre-defined types.
   a. For each needed MIME entry on the “MIME type collection” on page 178 page, click **New**.
   b. On the “MIME type settings” on page 178 page, specify a MIME type and extension.
   c. Click **OK** and **Save** to save your configuration change.
7. Regenerate the Web server plug-in configuration.
   a. **Click Servers > Web servers**, then select the appropriate Web server.
   b. Click **Generate Plug-in**, then click **Propagate Plug-in**.
8. Restart the application server.

Virtual host collection

Use this page to create and manage configurations that each let a single host machine resemble multiple host machines. Such configurations are known as *virtual hosts.*

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

Each virtual host has a logical name (which you define on this panel) and is known by its list of one or more domain name system (DNS) aliases. A DNS alias is the TCP/IP host name and port number used to request the servlet, for example yourHostName:80. (Port 80 is the default.)

You define one or more alias associations by clicking an existing virtual host or by adding a new virtual host.

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

An application server profile provides a default virtual host with some common aliases, such as the internet protocol (IP) address, the DNS short host name, and the DNS fully qualified host name. The alias comprises the first part of the path for accessing a resource such as a servlet.

For example, the alias is localhost:80 in the request http://localhost:80/myServlet.

A virtual host is not associated with a particular profile or node (machine), but is associated with a particular server instead. It is a configuration, rather than a ″live object.″ You can create a virtual host, but you cannot start or stop it.

For many users, creating virtual hosts is unnecessary because the default_host that is provided is sufficient.

Adding the host name and IP address of the localhost machine to the alias table lets a remote user access the administrative console.

Resources associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine.

Name

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

Virtual hosts enable you to isolate, and independently manage, multiple sets of resources on the same physical machine. Determine whether you need a virtual host alias for each port associated with an HTTP transport channel or an HTTP transport. There must be a virtual host alias corresponding to each port used by an HTTP transport channel or an HTTP transport. There is one HTTP transport channel or HTTP transport associated with each Web container, and there is one Web container in each application server.

When you create a virtual host, a default set of 90 MIME entries is created for the virtual host.

You must create a virtual host for each HTTP port in the following cases:

- You use the internal HTTP transport with a port other than the default value of 9080, or for some reason the virtual host does not contain the usual entry for port 9080.
- You create multiple application servers (stand-alone servers, managed servers, or cluster members) that are using the same virtual host. Because each server must be listening on a different HTTP port, you need a virtual host alias for the HTTP port of each server.

Virtual host settings

Use this page to configure a virtual host instance.

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

Name:

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

Data type	String
Default	default_host

Host alias collection

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

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

Host Name:

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

The product provides a default virtual host (named default_host). The virtual host configuration uses the wildcard character * (asterisk) along with the port number for its virtual host entries. Unless you specifically want to isolate resources from one another on the same node (physical machine), you probably do not need any virtual hosts in addition to the default host.

Port:

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

Host alias settings:

Use this page to view and configure a host alias.

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

Host name:

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

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

When there is no port number specified for a host alias, the default port is 80. For existing virtual hosts, the default host name and port reflect the values specified at product installation or configuration. For new virtual hosts, the default can be `*` to allow any value or no specification.

Data type	String
Default	*
	You can also use the IP address or the long or short DNS name.

Port:

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

The default reflects the value specified at product setup. The default might be 80, 81, 9060 or a similar value.

Data type Integer
Default 9060

MIME type collection

Use this page to view and configure multi-purpose internet mail extensions (MIME) object types and their file name extensions.

The list shows a collection of MIME type extension mappings defined for the virtual host. Virtual host MIME entries apply when you do not specify MIME entries at the Web module level.

To view a list of current virtual host Mime types in the administrative console, click **Environment > Virtual Hosts >** *virtual_host_name* **> MIME Types**.

MIME Type:

Specifies a MIME type, which can be application, audio, image, text, video, www, or x-world. An example value for MIME type is text/html.

Extensions:

Specifies file extensions of files that map the MIME type. Do not specify the period before the extension. Example extensions for a text/html MIME type are htm and html.

MIME type settings:

Use this page to configure a multi-purpose internet mail extensions (MIME) object type.

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

MIME Type:

Specifies a MIME type, which can be application, audio, image, text, video, www, or x-world. An example value for MIME type is text/html.

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

Data type String

Extensions:

Specifies file extensions of files that map the MIME type. Do not specify the period before the extension. Example extensions for a text/html MIME type are htm and html.

File extensions for a text/html MIME type are .htm and .html. A default value appears only if you are viewing the configuration for an existing MIME type.

Data type String

Variables

A variable is a configuration property that can be used to provide a parameter for some values in the system. A variable has a name and a value.

Not all WebSphere components support the use of a variable that you can define using this function. Test your application to verify that variables that you define are being used correctly.

WebSphere variables are used for:

- Configuring WebSphere Application Server path names, such as JAVA_HOME, and APP_INSTALL_ROOT.
- Configuring certain cell-wide or cluster-widecustomization values.

Each variable has a scope. A scope is the range of locations in the WebSphere Application Server network where the variable is applicable.

- A variable with a cell-wide scope is available across the entire deployment manager cell.
- A variable with a cluster-wide scope is available across the entire cluster in the cell.
- A variable with a node-level scope is available only on the node and the servers on that node. If a node-level variable has the same name as a cell-wide variable, the node-level variable value takes precedence.
- A server variable is available only on the one server process. A server variable takes precedence over a variable with the same name that is defined at a higher level.

You can use variables in configuration values such as file system path settings. Use the following syntax to refer to a variable:

`${variable_name}`

The value of a variable can contain a reference to another variable. The value of the variable is computed by substituting the value of the referenced variable recursively.

Variables are useful when concatenating two path variables when the specification does not accept the `AND` operator. For example, suppose that the following variables exist:

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

The variable reference `${USER_DIR}` resolves to the value `/home/myuserdir`.

Configuring WebSphere variables

This topic describes how to create a WebSphere Application Server variable.

You can define a WebSphere Application Server variable to provide a parameter for some values in the system. After you define the name and value for a variable, the value is used in place of the variable name. Variables most often specify file paths. However, some system components also support the use of variables. The Variable settings topic supplies further details about specifying variables and highlights further details about WebSphere Application Server components that use them.

WebSphere variables are used for:

- Configuring WebSphere Application Server path names, such as JAVA_HOME, and APP_INSTALL_ROOT.

- Configuring certain cell-wide or cluster-wide customization values.

The scope of a variable can be cell-wide, cluster-wide, node-wide, or applicable to only one server process.

Define variables on the **Environment > WebSphere Variables** console page.

Define the scope to apply a variable cell-wide, cluster-wide, node-wide, or to only one server process. A variable resolves to its new value when used in a component that supports the use of variables.

1. Click **Environment > WebSphere Variables** in the administrative console to define a new variable.
2. Specify the scope of the variable.

   Declare the new variable for the **Cell**, **Cluster**, **Node**, or **Server** and click **Apply**.

   The variable exists at the level you specify. Define a variable at multiple levels to use multiple values. The more granular definition overrides the higher level setting.

   For instance, if you specify the same variable on a node and a server, the server setting overrides the node setting. Similarly, a node level setting overrides a cluster or cell setting.

   Scoping variables is particularly important when testing data source objects. Variable scoping can cause a data source to fail the test connection, but to succeed at run time, or to pass the test connection, but fail at run time.

   See the *Administering applications and their environment* PDF for more information.
3. Click **New** on the WebSphere Variables page.
4. Specify a name, a value, and a description on the Variable page. Click **OK**.
5. Verify that the variable is displayed in the list of variables. The administrative console does not pick up typing errors. The variable is ignored if it is referred to incorrectly.
6. Save your configuration.
7. Stop the server and start the server again to put the variable configuration change into effect.

WebSphere variables collection

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

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

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

Name

Specifies the symbolic name for a WebSphere Application Server variable. For example, a variable name might represent a physical path or URL root used by WebSphere Application Server.

Value

Specifies the value that the symbolic name represents. For example, the value might be an absolute path value for a file or URL root.

Scope

Specifies the level at which a WebSphere Application Server variable is visible on the administrative console panel.

A resource can be visible in the administrative console collection table at the cell, cluster, node, or server scope.

Variable settings

Use this page to define the name and value of a WebSphere Application Server substitution variable.

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

Name:

Specifies the symbolic name for a WebSphere Application Server variable. For example, a variable name might represent a physical path or URL root that is used by WebSphere Application Server.

WebSphere Application Server variables are used for:

- Configuring WebSphere Application Server path names, such as *JAVA_HOME*, and APP_INSTALL_ROOT.
- Configuring certain cell-wide customization values.

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

For example, *JAVA_HOME* is the symbolic name representing the file system path to the installation directory for the Java virtual machine (JVM). For example, the value is /opt/IBM/WebSphere/AppServer/java for the WebSphere Application Server product on a Linux machine.

You can create new variables for use in WebSphere Application Server components that support the use of variables.

The WebSphere Application Server security component supports variables when establishing administrative security, but only the following:

- APP_INSTALL_ROOT
- WAS_INSTALL_ROOT
- USER_INSTALL_ROOT
- WAS_TEMP_DIR
- WAS_PROPS_DIR
- WAS_ETC_DIR

The security component uses these variables as security defaults when making substitutions that identify a path to the security configuration settings.

Data type String

Value:

Specifies the value that the symbolic name represents. For example, the value might be an absolute path value for a file or URL root.

For example, `/opt/IBM/WebSphere/AppServer/java` is the value on a Linux machine for a variable named JAVA_HOME.

Data type String

Description:

Documents the purpose of a variable.

Data type String

Variables

Variables in the WebSphere Application Server environment come in many varieties. They are used to control settings and properties relating to the server environment. Three main variable options that are important for a WebSphere Application Server user to know and understand are environment variables, and WebSphere Application Server variables, and custom properties.

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

WebSphere Application Server variables

WebSphere Application Server variables are used for three purposes:

- Configuring WebSphere Application Server path names, such as JAVA_HOME, and APP_INSTALL_ROOT
- Configuring certain cell-wide customization values

WebSphere Application Server variables are specified in the administrative console by clicking **Environment > Manage WebSphere variables**. How the variable is set determines its scope.

A variable can apply to a cell, a cluster, a node, or a server.

If the variable is set:

- At the server level, it applies to the entire server.
- At the node level, it applies to all servers in the node, unless you set the same variable at the server level. In that case, for that server, the setting that is specified at the server level overrides the setting that is specified at the node level.
- 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 that is specified at the server level overrides the setting that is specified at the cell level.
  - If you set the same variable at the node level, for all servers in that node, the setting that is specified at the node level overrides the setting that is 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. A limited number of supported custom properties are available and these properties can be set in the administrative console using the custom properties link that is associated with the functional component.

For example, to set Web container custom properties, click **Servers > Application Servers >** *server_name* **> Web Container Settings > Web container > Custom Properties**

Custom properties set from the Web container custom properties page apply to all transports that are associated with that Web container; custom properties set from one of the Web container transport chain or HTTP transport custom properties pages apply only to that specific HTTP transport chain or HTTP transport. If the same property is set on both the Web container page and either a transport chain or HTTP transport page, the settings on the transport chain or HTTP transport page override the settings that are defined for the Web container for that specific transport.

IBM Toolbox for Java JDBC driver

WebSphere Application Server supports the **IBM Toolbox for Java** JDBC driver. The IBM Toolbox for Java JDBC driver is included with the IBM Toolbox for Java product.

IBM Toolbox for Java is a library of Java classes that are optimized for accessing iSeries and AS/400 data and resources. You can use the IBM Toolbox for Java JDBC driver to access local or remote **DB2 UDB for iSeries** databases from server-side and client Java applications that run on any platform that supports Java.

IBM Toolbox for Java is available in these versions:

IBM Toolbox for Java licensed program

The licensed program is available with every OS/400 release, starting with Version 4 Release 2 (V4R2). You can install the licensed program on your iSeries system, and then either copy the IBM Toolbox for Java JAR file (*jt400.jar*) to your system or update your system *classpath* to locate the server installation. Product documentation for IBM Toolbox for Java is available from the iSeries information center: http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/index.htm Locate the documentation by traversing the following path in the left-hand navigation window of the iSeries information center: **Programming** > **Java** > **IBM Toolbox for Java**.

JTOpen

JTOpen is the open source version of IBM Toolbox for Java, and is more frequently updated than the licensed program version. You can download JTOpen from http://www-1.ibm.com/servers/eserver/iseries/toolbox/downloads.htm. You can also download the *JTOpen Programming Guide*. The guide includes instructions for installing JTOpen and information about the JDBC driver.

The JDBC driver for both versions supports JDBC 3.0. For more information about IBM Toolbox for Java and JTOpen, see the product Web site at http://www-1.ibm.com/servers/eserver/iseries/toolbox/index.html.

Note: If you are using WebSphere Application Server on platforms other than iSeries, use the **JTOpen** version of the Toolbox JDBC driver.

Configure and use the jt400.jar file

The following steps describe how to configure and utilize the jt400.jar file.

1. Download the *jt400.jar* file from the **JTOpen** URL at http://www-1.ibm.com/servers/eserver/iseries/toolbox/downloads.htm.

   Place it in a directory on your workstation such as */JDBC_Drivers/Toolbox*.
2. Open the administrative console.
3. Select **Environment > WebSphere Variables**.
4. Set the WebSphere variable *OS400_TOOLBOX_JDBC_DRIVER_PATH* at the **Node** level.
5. Double click **OS400_TOOLBOX_JDBC_DRIVER_PATH**.
6. Set the value to the full directory path to the `jt400.jar` file downloaded in step one. Do not include *jt400.jar* in this value.

   For example:

   ```
   OS400_TOOLBOX_JDBC_DRIVER_PATH == "/JDBC_Drivers/Toolbox"
   ```

   When you choose a Toolbox driver from the list of possible resource providers the **Classpath** field looks like:

   ```
   Classpath == ${OS400_TOOLBOX_JDBC_DRIVER_PATH}/jt400.jar
   ```

Shared library files

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

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

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

Managing shared libraries

Shared libraries are files used by multiple applications. Using the administrative console, you can define a shared library at the cell, node, or server level. You can then associate the library to an application, module or server to load the classes represented by the shared library in either a server-wide or application-specific class loader. Using an installed optional package, you can associate a shared library to an application by declaring the dependent library `.jar` file in the `MANIFEST.MF` file of the application. Refer to the Java 2 Platform, Enterprise Edition (J2EE) 1.4 specification, section 8.2 for an example.

If your deployed applications use shared library files, define shared libraries for the library files and associate the libraries with specific applications or modules or with an application server. Associating a shared library file with a server associates the file with all applications on the server. Use the Shared Libraries page to define new shared library files to the system and remove them.

- Use the administrative console to define a shared library.
  1. Create a shared library for each library file that your applications need.
  2. Associate each shared library with an application or module.
     - Associate a shared library with an application or module that uses the shared library file.
     - Associate a shared library with an application server so every application on the server can use the shared library file.
- Use an installed optional package to declare a shared library for an application.
- Remove a shared library.
  1. Click **Environment > Shared Libraries** in the console navigation tree to access the Shared Libraries page.
  2. Select the library to be removed.
  3. Click **Delete**.

  The list of shared libraries is refreshed. The library file no longer displays in the list.

Creating shared libraries

Shared libraries are files used by multiple applications.

The first step for making a library file available to multiple applications deployed on a server is to create a shared library for each library file that your applications need. When you create the shared libraries, set variables for the library files.

Use the Shared Libraries page to create and configure shared libraries.

1. Go to the Shared Libraries page.

   Click **Environment > Shared Libraries** in the console navigation tree.
2. Change the scope of the collection table to see what shared libraries are in a cell, node, server, or cluster.
   a. Select a cell, node, server, or cluster.
   b. Click **Apply**.

3. Click **New**.
4. Configure the shared library.
   a. On the settings page for a shared library, specify the name, class path, and any other variables for the library file that are needed.

      If the shared library specifies a native library path, refer to “Configuring native libraries in shared libraries.”
   b. Click **Apply**.
5. Repeat steps 1 through 4 until you define a shared library instance for each library file that your applications need.

Using the administrative console, associate your shared libraries with specific applications or modules or with the class loader of an application server. Associating a shared library file with a server class loader associates the file with all applications on the server.

Alternatively, you can use an installed optional package to associate your shared libraries with an application.

Configuring native libraries in shared libraries

Native libraries are platform-specific library files, including `.dll`, `.so`, or *SRVPGM objects, that can be configured within shared libraries. Native libraries are visible to an application class loader whenever the shared library is associated with an application. Similarly, native libraries are visible to an application server class loader whenever the shared library is associated with an application server.

When designing a shared library, consider the following conditions regarding Java native library support:

- The Java virtual machine (JVM) allows only one class loader to load a particular native library.
- There is no application programming interface (API) to unload a native library from a class loader.

  Native libraries are unloaded by the JVM when the class loader that found the library is collected from the heap during garbage collection.
- Application server class loaders persist for the duration of the application server.
- Application class loaders persist until an application is stopped or dynamically reloaded.

  If a shared library that is configured with a native library path is associated with an application, whenever the application is restarted or dynamically reloaded the application might fail with an UnsatisfiedLinkError indicating that the library is already loaded. The error occurs because, when the application restarts, it invokes the shared library class to reload the native library. The native library, however, is still loaded in memory because the application class loader which previously loaded the native library has not yet been garbage collected.
- Only the JVM class loader can load a dependent native library.

  For example, if *NativeLib1* is dependent on *NativeLib2*, then *NativeLib2* must be visible to the JVM class loader. The path containing *NativeLib2* must be specified on Java library path defined by the LIBPATH environment variable. If a native library configured in a shared library is dependent on other native libraries, the dependent libraries must be configured on the LIBPATH of the JVM hosting the application server in order for that library to load successfully.

When configuring a shared library on a shared library settings page, if you specify a value for **Native Library Path**, the native libraries on this path are not located by the WebSphere Application Server application or shared library class loaders unless the class which loads the native library was itself loaded by the same class loader.

Because a native library cannot be loaded more than once by a class loader, it is preferable for native libraries to be loaded within shared libraries associated with the class loader of an application server, because these class loaders persist for the lifetime of the server.

1. Implement a static method in the class that loads the native library.

In the class that loads the native library, call `System.loadLibrary(native_library)` in a static block. For example:

```
static {System.loadLibrary("native_library");
```

native_library loads during the static initialization of the class, which occurs exactly once when the class loads.

2. On the shared library settings page, set values for **Classpath** and **Native Library Path** that enable the shared library to load the native library.
3. Associate the shared library with the class loader of an application server.

   Associating a shared library with the class loader of an application server, rather than with an application, ensures that the shared library is loaded exactly once by the application server class loader, even though applications on the server are restarted or dynamically reloaded. Because the native library is loaded within a static block, the native library is never loaded more than once.

Shared library collection

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

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

Create a shared library for each library file that your application needs:

1. See what shared libraries are in a cell, node, or server.

   By default, a shared library is accessible to applications deployed (or installed) on the same node as the shared library file. Use the **Scope** field to change the scope to a different node or to a specific server. Under **Scope**, select the cell, a node, or a server and click **Apply**. Changing the scope of the collection table enables you to find shared libraries.
2. Select the scope for your new shared library.
3. Click **New**.
4. On the settings page for the new shared library, specify the name, class path, and any other variables for the library file that are needed.

After you create a shared library, associate it with an application or module or with the class loader of a server:

- To associate a shared library with an application or module, click **Applications > Enterprise Applications >** *application_name* **> Shared library references**. On the Shared library references page for the application, select the shared library file and click **OK**.
- To associate a shared library with a server class loader, click **Servers > Application servers >** *server_name* **> Java and Process Management > Class loader >** *class_loader_ID* **> Shared library references >** *shared_library_name*. On the settings page for the library reference for the server class loader, specify values that identify the shared library file.

Name

Specifies a name for the shared library.

Description

Describes the shared library file.

Shared library settings

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

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

Scope:

Specifies whether the shared library has its configuration file in a location that pertains to the cell, node, server, or cluster level.

Data type String

Name:

Specifies a name for the shared library.

Data type String

Description:

Describes the shared library file.

Data type String

Classpath:

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

Data type String
Units Class path

Native Library Path:

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

If you specify a value for **Native Library Path**, the native libraries are not located by the WebSphere Application Server application or shared library class loaders unless the following conditions exist:

- A class loads the native libraries.
- The application invokes a method in this class which loads the libraries.

  For example, in the class that loads the native library, call `System.loadLibrary(`*`native_library`*`)` in a static block:

  ```
  static {System.loadLibrary("native_library");
  ```
- The **Classpath** specified on this page contains the class that loads the libraries.

Native libraries cannot be loaded more than once by a class loader. Thus, it is preferable for native libraries to be loaded within shared libraries associated with the class loader of an application server.

Data type String
Units Class path

Associating shared libraries with applications or modules

You can associate a shared library with an application or module. Classes represented by the shared library are then loaded in the application's class loader, making the classes available to the application.

This topic assumes that you have defined a shared library at the cell, node, server, or cluster level. The shared library represents a library file used by multiple deployed applications.

This topic also assumes that you want to use the administrative console, and not an installed optional package, to associate a shared library with an application.

To associate a shared library with an application or module, create and configure a library reference using the administrative console. A library reference specifies the name of the shared library file.

If you associate a shared library with an application, do not associate the same shared library with a server class loader.

1. Click **Applications > Enterprise Applications >** *application_name* **> Shared library references** in the console navigation tree to access the Shared library references page.
2. On the Shared library references page, select an application or module to which you want to associate a shared library.
3. Click **Reference shared libraries**.
4. On the Shared Library Mapping page, select one or more shared libraries that the application or modules uses in the **Available** list, click **>>** to add them to the **Selected** list, and click **OK**.
5. Repeat steps 2 through 4 until you define a library reference instance for each shared library that your application or module requires.
6. On the Shared library references page, click **OK**.

When you run the application, classes represented by the shared library are loaded in the application's class loader, making the classes available to the application or module.

Shared library reference and mapping settings

Use the Shared library references and Shared Library Mapping pages to associate defined shared libraries with an application or Web module. A shared library is an external Java archive (JAR) file that is used by one or more applications. Using shared libraries enables multiple applications deployed on a server to use a single library, rather than use multiple copies of the same library. After you associate shared libraries with an application or module, the application or module class loader loads classes represented by the shared libraries and makes those classes available to the application or module.

To view the Shared library references console page, click **Applications > Enterprise Applications >** *application_name* **> Shared library references**. To view the Shared Library Mapping page, click **Reference shared libraries** on the Shared library references page. These pages are the same as the Shared Library Mapping for Modules and Shared Library Mapping pages in the application installation and update wizards.

On the Shared library references page, the first element listed is the application. The other elements are modules in the application.

To associate shared libraries with your application or module:

1. Select an application or module.
2. Click **Reference shared libraries**.
3. On the Shared Library Mapping page, select one or more shared libraries that the application or modules uses in the **Available** list, click **>>** to add them to the **Selected** list, and click **OK**.

A defined shared library for a file that your application or module uses must exist to associate your application or module to the library.

If no shared libraries are defined and the application is installed already, on the Shared Library Mapping page, click **New** and define a shared library.

You can otherwise define a shared library as follows:

1. Click **Environment > Shared Libraries**.

2. Specify whether the shared library is visible at the cell, node or server level.
3. Click **New**.
4. On the settings page for the new shared library, specify a name and one or more class paths. If the libraries are platform-specific files such as `.dll`, `.so`, or *SRVPGM objects, also specify a native library path. Then, click **Apply**.
5. Save the administrative configuration.

Application:

Specifies the name of the application that you are installing or that you selected on the Enterprise Applications page.

Module:

Specifies the name of the module associated with the shared libraries.

URI:

Specifies the location of the module relative to the root of the application EAR file.

Shared libraries:

Specifies the name of the shared library files associated with the application or module.

Associating shared libraries with servers

You can associate shared libraries with the class loader of a server. Classes represented by the shared library are then loaded in a server-wide class loader, making the classes available to all applications deployed on the server.

This topic assumes that you have defined a shared library at the cell, node, server, or cluster level. The shared library represents a library file used by multiple deployed applications.

To associate a shared library with the class loader of a server, create and configure a library reference using the administrative console. A library reference specifies the name of the shared library file.

If you associate a shared library with a server class loader, do not associate the same shared library with an application.

1. Configure class loaders for applications deployed on the server.
   a. Click **Servers > Application Servers > *server_name*** to access the settings page for the application server.
   b. Set values for the application **Class loader policy** and **Class loading mode** of the server.

      For information on these settings, see “Application server settings” on page 213 and the *Administering applications and their environment* PDF.
2. Create a library reference for each shared library file that your application needs.
   a. In the administrative console, click **Servers > Application servers >** *server_name* **> Java and Process Management > Class loader >** *class_loader_ID*.
   b. Click **Shared library references** to access the Library Reference page.
   c. Click **Add**.
   d. On the settings page for a library reference, name the library reference. The name identifies the shared library file that your application uses.
   e. Click **Apply**. The name of the library reference is shown in the list on the Library Reference page.

Repeat the previous steps until you define a library reference for each shared library that your application needs.

Installed optional packages

Installed optional packages enable applications to use the classes in Java archive (`.jar`) files without having to include them explicitly in a class path. An installed optional package is a `.jar` file containing specialized tags in its manifest file that enable the application server to identify it. An installed optional package declares one or more shared library `.jar` files in the manifest file of an application. When the application is installed on a server or cluster, the classes represented by the shared libraries are loaded in the class loader of the application, making the classes available to the application.

When a Java 2 Platform, Enterprise Edition (J2EE) application is installed on a server or cluster, dependency information is specified in its manifest file. WebSphere Application Server reads the dependency information of the application (`.ear` file) to automatically associate the application with an installed optional package `.jar` file. WebSphere Application Server adds the `.jar` files in associated optional packages to the application class path. Classes in the installed optional packages are then available to application classes.

Installed optional packages used by WebSphere Application Server are described in section 8.2 of the J2EE specification, Version 1.4 at http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf.

WebSphere Application Server supports using the manifest file (`manifest.mf`) in shared library `.jar` files and application `.ear` files. WebSphere Application Server does not support the Java 2 Platform Standard Edition (J2SE) Installed Optional Package semantics used in the J2SE specification (http://java.sun.com/j2se/1.3/docs/guide/extensions/spec.html), which primarily serve the applet environment. WebSphere Application Server ignores applet-specific tags within manifest files.

Sample manifest.mf file

A sample manifest file follows for an application `app1.ear` that refers to a single shared library file `util.jar`:

```
app1.ear:
    META-INF/application.xml
    ejb1.jar:
         META-INF/MANIFEST.MF:
             Extension-List: util
             util-Extension-Name: com/example/util
             util-Specification-Version: 1.4
         META-INF/ejb-jar.xml

util.jar:
    META-INF/MANIFEST.MF:
        Extension-Name: com/example/util
        Specification-Title: example.com's util package
        Specification-Version: 1.4
        Specification-Vendor: example.com
        Implementation-Version: build96
```

The syntax of a manifest entry depends on whether the entry applies to a member with a defining role (the shared library) or a member with a referencing role (a J2EE application or a module within a J2EE application).

Manifest entry tagging

Main tags used for manifest entries include the following:

Extension-List
A required tag with variable syntax. Within the context of the referencing role (application's

manifest), this is a space delimited list that identifies and constructs unique Extension-Name, Extension-Specification tags for each element in the list. Within the context of the defining role (shared library), this tag is not valid.

Extension-Name
A required tag that provides a name and links the defining and referencing members. The syntax of the element within the referencing role is to prefix the element with the `<ListElement>` string. For each element in the Extension-List, there is a corresponding `<ListElement>`-Extension-Name tag. The defining string literal value for this tag (in the above sample `com/example/util1`) is used to match (in an equality test) the corresponding tags between the defining and referencing roles.

Specification-Version
A required tag that identifies the specification version and links the defining and referencing members.

Implementation-Version
An optional tag that identifies the implementation version and links the defining and referencing members.

Further information on these tags is in the `.jar` file specification at http://java.sun.com/j2se/1.4.2/docs/guide/jar/jar.html#Manifest%20Specification.

Using installed optional packages

You can associate one or more shared libraries with an application using an installed optional package that declares the shared libraries in the application's manifest file. Classes represented by the shared libraries are then loaded in the application's class loader, making the classes available to the application.

Read about installed optional packages in "Installed optional packages" on page 190 and in section 8.2 of the Java 2 Platform, Enterprise Edition (J2EE) specification, Version 1.4 at http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf.

WebSphere Application Server does not support the Java 2 Platform Standard Edition (J2SE) Installed Optional Package semantics used in the J2SE specification (http://java.sun.com/j2se/1.3/docs/guide/extensions/spec.html), which primarily serve the applet environment. WebSphere Application Server ignores applet-specific tags within manifest files.

Installed optional packages expand the existing shared library capabilities of an application server. Prior to Version 6, an administrator was required to associate a shared library to an application or server. Installed optional packages enable an administrator to declare a dependency in an application's manifest file to a shared library, with installed optional package elements listed in the manifest file, and automatically associate the application to the shared library. During application installation, the shared library `.jar` file is added to the class path of the application class loader.

If you use an installed optional package to associate a shared library with an application, do not associate the same shared library with an application class loader or a server class loader using the administrative console.

1. Assemble the library file, including the manifest information that identifies it as an extension. Two sample manifest files follow. The first sample manifest file has application `app1.ear` refer to a single shared library file `util.jar`:

```
app1.ear:
    META-INF/application.xml
    ejb1.jar:
         META-INF/MANIFEST.MF:
             Extension-List: util
             util-Extension-Name: com/example/util
             util-Specification-Version: 1.4
         META-INF/ejb-jar.xml
```

```
util.jar:
    META-INF/MANIFEST.MF:
        Extension-Name: com/example/util
        Specification-Title: example.com's util package
        Specification-Version: 1.4
        Specification-Vendor: example.com
        Implementation-Version: build96
```

The second sample manifest file has application `app1.ear` refer to multiple shared library `.jar` files:

```
app1.ear:
    META-INF/application.xml
    ejb1.jar:
         META-INF/MANIFEST.MF:
             Extension-List: util1 util2 util3
             Util1-Extension-Name: com/example/util1
             Util1-Specification-Version: 1.4
             Util2-Extension-Name: com/example/util2
             Util2-Specification-Version: 1.4
             Util3-Extension-Name: com/example/util3
             Util3-Specification-Version: 1.4
         META-INF/ejb-jar.xml

util1.jar:
    META-INF/MANIFEST.MF:
        Extension-Name: com/example/util1
        Specification-Title: example.com's util package
        Specification-Version: 1.4
        Specification-Vendor: example.com
        Implementation-Version: build96

util2.jar:
    META-INF/MANIFEST.MF:
        Extension-Name: com/example/util2
        Specification-Title: example.com's util package
        Specification-Version: 1.4
        Specification-Vendor: example.com
        Implementation-Version: build96

util3.jar:
    META-INF/MANIFEST.MF:
        Extension-Name: com/example/util3
        Specification-Title: example.com's util package
        Specification-Version: 1.4
        Specification-Vendor: example.com
        Implementation-Version: build96
```

2. Create a shared library that represents the library file assembled in step 1. This installs the library file as a WebSphere Application Server shared library.
3. Copy the shared library `.jar` file to the cluster members.
4. Assemble the application, declaring in the application manifest file dependencies to the library files named the manifest created for step 1.

   See the *Developing and deploying applications* PDF for more information.
5. Install the application on the server or cluster.

   See the *Developing and deploying applications* PDF for more information.

During application installation, the shared library `.jar` files are added to the class path of the application class loader.

Library reference collection

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

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Java and Process Management > Class loader >** *class_loader_ID* **> Shared library references**.

If no shared libraries are defined in your environment, such as at the node or server scope, after you click **Add** a message is displayed stating that you must define a shared library before you can create a library reference. A shared library is a container-wide library file that can be used by deployed applications. To define a shared library, click **Environment > Shared Libraries** and specify the scope of the container. Then, click **New** and specify a name and one or more paths for the shared library. After you define a shared library, return to this page, click **Add**, and create a library reference.

Library name
Specifies a name for the library reference.

Library reference settings
Use this page to define library references, which specify how to use global libraries.

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Java and Process Management > Class loader >** *class_loader_ID* **> Shared library references >** *library_reference_name*.

A shared library is a container-wide library file that can be used by deployed applications. To define a shared library, click **Environment > Shared Libraries** and specify the scope of the container. Then, click **New** and specify a name and one or more paths for the shared library.

Library name:

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

Data type String

Environment: Resources for learning

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

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

Programming instructions and examples
- WebSphere Application Server education

Administration
- Listing of all IBM WebSphere Application Server Redbooks

Chapter 6. Working with server configuration files

This topic show how to manage application server configuration files.

Application server configuration files define the available application servers, their configurations, and their contents.

You should periodically save changes to your administrative configuration. You can change the default locations of configuration files, as needed.

- Edit configuration files.

  The master repository is comprised of .xml configuration files

  You can edit configuration files using

  – The administrative console. See the Using the administrative console topic in the *Using the administrative clients* PDF.
  – Scripting. See the Getting started with scripting topic in the *Using the administrative clients* PDF.
  – The wsadmin commands. See the Using command line tools topic in the *Using the administrative clients* PDF.
  – Programing. See the Using administrative programs (JMX) topic in the *Using the administrative clients* PDF.
  – By editing a configuration file directly.
- Save changes made to configuration files. Using the console, you can save changes as follows:
  1. In the navigation select **System Administration > Save changes to master repository**.
  2. Put a check mark in the **Synchronize changes with Nodes** check box.
  3. Click **Save**.
- Handle temporary configuration files resulting from a session timing out.
- Change the location of temporary configuration files.
- Change the location of backed-up configuration files.
- Change the location of temporary workspace files.
- Back up and restore configurations.

Configuration documents

WebSphere Application Server stores configuration data for servers in several documents in a cascading hierarchy of directories. The configuration documents describe the available application servers, their configurations, and their contents. Most configuration documents have XML content.

Hierarchy of directories of documents

The cascading hierarchy of directories and the documents' structure support multinode replication to synchronize the activities of all servers in a cell. In a Network Deployment environment, changes made to configuration documents in the cell repository, are automatically replicated to the same configuration documents that are stored on nodes throughout the cell.

At the top of the hierarchy is the **cells** directory. It holds a subdirectory for each cell. The names of the cell subdirectories match the names of the cells. For example, a cell named *cell1* has its configuration documents in the subdirectory *cell1*. The name of the cell must be different from the cluster name pair.

On the Network Deployment node, the subdirectories under the cell contain the entire set of documents for every node and server throughout the cell. On other nodes, the set of documents is limited to what applies

to that specific node. If a configuration document only applies to *node1*, then that document exists in the configuration on *node1* and in the Network Deployment configuration, but not on any other node in the cell.

Each cell subdirectory has the following files and subdirectories:

- The `cell.xml` file, which provides configuration data for the cell
- Files such as `security.xml`, `virtualhosts.xml`, `resources.xml`, and `variables.xml`, which provide configuration data that applies across every node in the cell
- The **clusters** subdirectory, which holds a subdirectory for each cluster defined in the cell. The names of the subdirectories under clusters match the names of the clusters.

  Each cluster subdirectory holds a cluster.xml file, which provides configuration data specifically for that cluster.
- The **nodes** subdirectory, which holds a subdirectory for each node in the cell. The names of the nodes subdirectories match the names of the nodes.

  Each node subdirectory holds files such as `variables.xml` and `resources.xml`, which provide configuration data that applies across the node. Note that these files have the same name as those in the containing cell's directory. The configurations specified in these node documents override the configurations specified in cell documents having the same name. For example, if a particular variable is in both cell- and node-level `variables.xml` files, all servers on the node use the variable definition in the node document and ignore the definition in the cell document.

  Each node subdirectory holds a subdirectory for each server defined on the node. The names of the subdirectories match the names of the servers. Each server subdirectory holds a `server.xml` file, which provides configuration data specific to that server. Server subdirectories might hold files such as `security.xml`, `resources.xml` and `variables.xml`, which provide configuration data that applies only to the server. The configurations specified in these server documents override the configurations specified in containing cell and node documents having the same name.
- The **applications** subdirectory, which holds a subdirectory for each application deployed in the cell. The names of the applications subdirectories match the names of the deployed applications.

  Each deployed application subdirectory holds a `deployment.xml` file that contains configuration data on the application deployment. Each subdirectory also holds a **META-INF** subdirectory that holds a J2EE application deployment descriptor file as well as IBM deployment extensions files and bindings files. Deployed application subdirectories also hold subdirectories for all .war and entity bean .jar files in the application. Binary files such as .jar files are also part of the configuration structure.

An example file structure is as follows:

```
cells
  cell1
     cell.xml resources.xml virtualhosts.xml variables.xml security.xml
     nodes
        nodeX
           node.xml variables.xml resources.xml serverindex.xml
           serverA
              server.xml variables.xml
           nodeAgent
              server.xml variables.xml
        nodeY
           node.xml variables.xml resources.xml serverindex.xml
     applications
        sampleApp1
           deployment.xml
           META-INF
              application.xml ibm-application-ext.xml ibm-application-bnd.xml
        sampleApp2
           deployment.xml
           META-INF
              application.xml ibm-application-ext.xml ibm-application-bnd.xml
```

Changing configuration documents

You can use one of the administrative tools (console, wsadmin, Java APIs) to modify configuration documents or edit them directly. It is preferable to use the administrative console because it validates changes made to configurations. ″“Configuration document descriptions”″ states whether you can edit a document using the administrative tools or must edit it directly.

Configuration document descriptions

Most configuration documents have XML content. The table describes the documents and states whether you can edit them using an administrative tool or must edit them directly.

If possible, edit a configuration document using the administrative console because it validates any changes that you make to configurations. You can also use one of the other administrative tools (wsadmin or Java APIs) to modify configuration documents. Using the administrative console or wsadmin scripting to update configurations is less error prone and likely quicker and easier than other methods.

However, you cannot edit some files using the administrative tools. Configuration files that you must edit manually have an X in the **Manual editing required** column in the table below.

Document descriptions

(The paths in the Locations column are split on multiple lines for publishing purposes.)

Configuration file	Locations	Purpose	Manual editing required
admin-authz.xml	config/cells/ *cell_name*/	Define a role for administrative operation authorization.	X
app.policy	config/cells/ *cell_name*/ nodes/*node_name*/	Define security permissions for application code.	X
cell.xml	config/cells/ *cell_name*/	Identify a cell.	
cluster.xml	config/cells/ *cell_name*/ clusters/ *cluster_name*/	Identify a cluster and its members and weights. This file is only available with the Network Deployment product.	
deployment.xml	config/cells/ *cell_name*/ applications/ *application_name*/	Configure application deployment settings such as target servers and application-specific server configuration.	
filter.policy	config/cells/ *cell_name*/	Specify security permissions to be filtered out of other policy files.	X
integral-jms-authorizations.xml	config/cells/ *cell_name*/	Provide security configuration data for the integrated messaging system.	X
library.policy	config/cells/ *cell_name*/ nodes/*node_name*/	Define security permissions for shared library code.	X
multibroker.xml	config/cells/ *cell_name*/	Configure a data replication message broker.	

namestore.xml	config/cells/ *cell_name*/	Provide persistent name binding data.	X
naming-authz.xml	config/cells/ *cell_name*/	Define roles for a naming operation authorization.	X
node.xml	config/cells/ *cell_name*/ nodes/*node_name*/	Identify a node.	
pmirm.xml	config/cells/ *cell_name*/	Configure PMI request metrics.	X
resources.xml	config/cells/ *cell_name*/ config/cells/ *cell_name*/ nodes/*node_name*/ config/cells/ *cell_name*/ nodes/*node_name*/ servers/ *server_name*/	Define operating environment resources, including JDBC, JMS, JavaMail, URL, JCA resource providers and factories.	
security.xml	config/cells/ *cell_name*/	Configure security, including all user ID and password data.	
server.xml	config/cells/ *cell_name*/ nodes/ *node_name*/ servers/ *server_name*/	Identify a server and its components.	
serverindex.xml	config/cells/ *cell_name*/ nodes/ *node_name*/	Specify communication ports used on a specific node.	
spi.policy	config/cells/ *cell_name*/ nodes/ *node_name*/	Define security permissions for service provider libraries such as resource providers.	X
variables.xml	config/cells/ *cell_name*/ config/cells/ *cell_name*/ nodes/ *node_name*/ config/cells/ *cell_name*/ nodes/*node_name*/ servers/ *server_name*/	Configure variables used to parameterize any part of the configuration settings.	
virtualhosts.xml	config/cells/ *cell_name*/	Configure a virtual host and its MIME types.	

Object names: What the name string cannot contain

When you create a new object using the administrative console or a wsadmin command, you often must specify a string for a name attribute.

Most characters are allowed in the name string. However, the name string cannot contain the following characters. The name string also cannot contain leading and trailing spaces.

/	forward slash
****	backslash
*****	asterisk
,	comma
:	colon
;	semi-colon
=	equal sign
+	plus sign
?	question mark
|	vertical bar
<	left angle bracket
>	right angle bracket
&	ampersand (and sign)
%	percent sign
'	single quote mark
"	double quote mark
]]>	No specific name exists for this character combination.
.	period (not valid if first character; valid if a later character)
#	Hash mark
$	Dollar sign

Configuration repositories

A configuration repository stores configuration data.

By default, configuration repositories reside in the *config* subdirectory of the profile root directory.

A cell-level repository stores configuration data for the entire cell and is managed by a file repository service that runs in the deployment manager. The deployment manager and each node have their own repositories. A node-level repository stores configuration data that is needed by processes on that node and is accessed by the node agent and application servers on that node.

When you change a WebSphere Application Server configuration by creating an application server, installing an application, changing a variable definition or the like, and then save the changes, the cell-level repository is updated. The file synchronization service distributes the changes to the appropriate nodes.

Handling temporary configuration files resulting from session timeout

If the console is not used for 15 minutes or more, the session times out. The same thing happens if you close the browser window without saving the configuration file. Changes to the file are saved to a temporary file when the session times out, after 15 minutes. This topic discusses what happens depending on whether you load the saved file.

A configuration file must have been saved from a previous administrative console session for the user ID that you are currently using to access the administrative console.

When a session times out, the configuration file in use is saved under the `userid/timeout` directory under the ServletContext's temp area. This value is the value of the javax.servlet.context.tempdir attribute of the ServletContext context. By default, it is: *profile_root*`/temp/hostname/Administration/admin/admin.war`

You can change the temp area by specifying it as a value for the tempDir init-param of the action servlet in the deployment descriptor (web.xml) of the administrative application.

The configuration file is also saved automatically when the same user ID logs into the non-secured console again, effectively starting a different session. This process is equivalent to forcing the existing user ID out of session, similar to a session timing out.

The next time you log on to the administrative console, you are prompted to load the saved configuration file. Do one of the following actions:

- Load the saved file.
  1. If a file with the same name exists in the *profile_root*`/config` directory, that file is moved to the `userid/backup` directory in the temp area.
  2. The saved file is moved to the *profile_root*`/config` directory.
  3. The file is then loaded.
- Do not load the saved file.

  The saved file is deleted from the `userid/timeout` directory in the temp area.

You loaded the saved configuration file if you chose to do so.

Once you have logged into the administrative console, do whatever administration of WebSphere Application Server that you need to do.

Changing the location of temporary configuration files

You can change the default directory where temporary configuration files are stored.

The configuration repository uses copies of configuration files and temporary files while processing repository requests. It also uses a backup directory while managing the configuration. You can change the default locations of these files from the configuration directory to a directory of your choice by using the administrative console.

The default location for the configuration temporary directory is *profile_root*/config/temp. Use the administrative console to change the location of the temporary repository file location for all types of server processes. For example, to change the setting for Application Server, do the following steps:

1. Click **Servers > Application servers** in the navigation tree of the administrative console. Then, click ***server name* > Administration > Administration services > Repository service > Custom properties**.
2. On the Properties page, click **New**.
3. On the settings page for a property, define a property for the temporary file location. The key for this property is `was.repository.temp`. The value is the full path name to the desired location.
4. Click **OK**.

Changing the location of backed-up configuration files

You can change the default directory where backup files are stored.

During administrative processes like adding a node to a cell or updating a file, configuration files are temporarily backed up to a backup location.

The default location for the backup configuration directory is *profile_root*/config/backup. Use the administrative console to change the location of the repository backup directory for all types of server processes. For example, to change the setting for Application Server, do the following steps:

1. Click **Servers > Application servers** in the navigation tree of the administrative console. Then, click ***server name*** **> Administration > Administration services > Repository service > Custom properties**.
2. On the Properties page, click **New**.
3. On the settings page for a property, define a property for the backup file location. The key for this property is `was.repository.backup`. The value is the full path name to the desired location.
4. Click **OK**.

Changing the location of temporary workspace files

The default workspace root is calculated based on the user installation root. This topic discusses how to change the location of temporary workspace files.

You must first install WebSphere Application Server before you change the location of temporary workspace files.

With the administrative console workspace, client applications can navigate the configuration. Each workspace has its own repository location defined either in the system property or the property that is passed to a workspace manager when creating the workspace: workspace.user.root or workspace.root, which is calculated as workspace.root/*user_ID*/workspace/wstemp.

The default workspace root is calculated based on the user installation root: *profile_root*/wstemp. You can change the default location of temporary workspace files:

Change the setting for the Java system property *workspace.user.root* or *workspace.root* so its value is no longer set to the default location.

Set the Java system property when launching a Java process using the -D option. For example, to set the default location to the full path of the root of all users' directories, use the following option:

`-Dworkspace.user.root=`*`full_path_for_root_of_all_user_directories`*

You changed the location of temporary workspace files.

Backing up and restoring administrative configuration files

This topic discusses how to back up and restore administrative configuration files.

WebSphere Application Server represents its administrative configurations as XML files. You should back up configuration files on a regular basis.

Restore the configuration only if the configuration files that you backed up are at the same level of the release, including fixes, as the release to which you are restoring.

1. Synchronize administrative configuration files.
   a. Click **System Administration > Nodes** in the console navigation tree to access the Nodes page.
   b. Click **Full Resynchronize**. The resynchronize operation resolves conflicts among configuration files and can take several minutes to run.
2. Run the backupConfig command to back up configuration files. See the backupConfig command topic in the *Using the administrative clients* PDF for information.

3. Run the restoreConfig command to restore configuration files. See the restoreConfig command topic in the *Using the administrative clients* PDF for information. Specify backup files that do not contain invalid or inconsistent configurations.

Backing up and recovering administrative configurations

Most of the configuration for your WebSphere Application Server instance resides in the `config` directory structure. In addition, the `properties` directory also contains several important configuration files. This topic discusses how to back up and recover your configuration.

This topic assumes familiarity with the `SAV` and `RST` commands.

The administrative configuration contains vital information regarding your WebSphere Application Server setup. Back up configuration files on a regular basis.

- Issue the `SAV` command to save configuration files and properties files.

  For example:

  ```
  SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file') OBJ(('profile_root/properties/*') ('profile_root/config/*'))
  ```

  The default instance name is default.

  The command is split for printing purposes. Enter the command on one command line.
- Issue the `RST` command to restore configuration files and properties files.

  For example:

  ```
  RST DEV('/QSYS.lib/wsalib.lib/wsasavf.file') OBJ('/QIBM/*')
  ```

You have saved and backed up your WebSphere Application Server configuration.

Transformation of configuration files

The WebSphere Application Server master configuration repository stores configuration files for all the nodes in the cell. When you upgrade the deployment manager from one release of WebSphere Application Server to another, the configuration files that are stored in the master repository for the nodes on the old release are converted into the format of the new release.

With this conversion, the deployment manager can process the configuration files uniformly. However, nodes on an old release cannot readily use configuration files that are in the format of the new release. WebSphere Application Server addresses the problem when it synchronizes the configuration files from the master repository to a node on an old release. The configuration files are first transformed into the old release format before they ship to the node. WebSphere Application Server performs the following transformations on configuration documents:

- Changes the XML name space from the format of the new release to the format of the old release
- Strips out attributes of cell-level documents that are applicable to the new release only
- Strips out new resource definitions that are not understood by old release nodes

Server configuration files: Resources for learning

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

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

View links to additional information:

Administration

- IBM WebSphere Application Server Redbooks

  This site contains a listing of all WebSphere Application Server Redbooks.
- IBM WebSphere developerWorks

  This site is the home of technical information for developers working with WebSphere products. You can download WebSphere software, take a fast path to developerWorks zones, such as VisualAge Java or WebSphere Application Server, learn about WebSphere products through a newcomers page, tutorials, technology previews, training, and Redbooks, get answers to questions about WebSphere products, and join the WebSphere community, where you can keep up with the latest developments and technical papers.
- WebSphere Application Server Support page

  Take advantage of the Web-based Support and Service resources from IBM to quickly find answers to your technical questions. You can easily access this extensive Web-based support through the IBM Software Support portal at URL `http://www-3.ibm.com/software/support/` and search by product category, or by product name. For example, if you are experiencing problems specific to WebSphere Application Server, click **WebSphere Application Server** in the product list. The WebSphere Application Server Support page appears.

Chapter 7. Administering application servers

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

Install WebSphere Application Server.

After you install WebSphere Application Server you might have to perform some of the following tasks. Other then creating application servers, these tasks can be performed in any order.

- Create application servers.
- Create clusters.
- Configure transport chains.
- Develop custom services.
- Define processes for the application server.
- Configure the Java virtual machine.

After preparing a server, deploy an application or component on the server. See the *Administering applications and their environment* PDF for a sample procedure that you might follow in configuring the application server runtime and resources.

Manage the application servers you created.

Application servers

Application servers extend the ability of a Web server to handle Web application requests, typically using Java technology. An application server makes it possible for a server to generate a dynamic, customized response to a client request.

For example, given:

1. A user at a Web browser on the Internet visits a company Web site. The user requests to use an application that provides access to data in a database.
2. The user request flows to the Web server.
3. The Web server determines that the request involves an application containing resources not handled directly by the Web server (such as servlets). It forwards the request to WebSphere Application Server.
4. WebSphere Application Server forwards the request to one of its application servers on which the application is running.
5. The invoked application then processes the user request. For example:
   - An application servlet prepares the user request for processing by an enterprise bean that performs the database access.
   - The application produces a dynamic Web page containing the results of the user query.
6. The application server collaborates with the Web server to return the results to the user at the Web browser.

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

Manage the WebSphere Application Server subsystem

WebSphere Application Server runs in its own subsystem, called QWAS61, which is installed with the product. For each WebSphere Application Server profile, the number of jobs that run in the QWAS61 subsystem depends on which product and which services you are using. For each Network Deployment profile, one deployment manager job is running in the QWAS61 subsystem.

A WebSphere Application Server administrator can perform the following tasks within this subsystem:

- Prepare the subsystem to run WebSphere Application Server profiles.
- Start the WebSphere Application Server environment.
- Start and stop generic application servers.
- Configure application servers to start automatically.
- **Optional:** Set explicit authorities for the **startServer** and the **stopServer** scripts. You can grant explicit authorities so that user profiles that do not have *ALLOBJ authority can run the **startServer** and **stopServer** scripts.
- Shut down the WebSphere Application Server subsystem.

Starting the WebSphere Application Server environment

WebSphere Application Server runs in its own subsystem, called QWAS61, which is installed with the product. Use these steps to prepare your system to run WebSphere Application Server profiles.

WebSphere Application Server profiles run in the QWAS61 subsystem and require that TCP/IP is configured and activated on the system. To prepare your system to run WebSphere Application Server profiles:

1. Start Transmission Control Protocol/Internet Protocol (TCP/IP). On the CL command line, enter:
   ```
   STRTCP
   ```
2. Start your application server profile. If the QWAS61 subsystem is not active, it is started when you start your application server profile.

   Important: The default application server does not automatically start when the subsystem is started. When you use the startServer Qshell script to start your server, the QWAS61 subsystem is started if it is not currently active.

Use one of the following methods to verify that the application server is running:

- On the CL command line, enter the Work with Active Jobs (WRKACTJOB) command:
  ```
  WRKACTJOB SBS(QWAS61)
  ```

  The SERVER1 job should be listed if WebSphere Application Server Version 6.1 is installed. When the server is ready to accept requests, the job log should contain message WAS0106, ″WebSphere application server *serverName* ready.″
- Look for QIBM_WSA_ADMIN under Server Jobs in the Operations Navigator.

Configuring application servers to automatically start when the QWAS61 subsystem starts

WebSphere Application Server runs in its own subsystem, called QWAS61, which is installed with the product. Use these steps to configure application servers to start automatically when the QUAS61 subsystem starts.

1. Give your user profile authority to the QWAS61/QWASJOBD job description and QWAS61/QWAS61 subsystem description. For each profile:
   a. Create a duplicate of the job description used by WebSphere Application Server profiles. For example, issue the following command on the CL command line:
      ```
      CRTDUPOBJ OBJ(QWASJOBD) FROMLIB(QWAS61) OBJTYPE(*JOBD) TOLIB(mywasjobd)
          NEWOBJ(myserv)
      ```
   b. Use the CHGJOBD command to change the newly created job description such that the Request data or command (RQSDTA) field starts the new server. For example, to start the default profile's application server (server1) when the subsystem is started, set the RQSDTA field as follows:

```
'QSYS/CALL PGM(QWAS61/QWASSTRSVR) PARM(''-profilePath''
       ''user_data_root/default''
       ''-server'' ''server1'')'
```

Important: If a stand-alone application server has been federated, or added, to a Network Deployment cell, specify the nodeagent server. If the node agent is not active, application servers cannot start.

2. Configure an application server so that it is automatically started by the node agent. To configure application servers in your Network Deployment cell to start automatically when you start the node agent for the node that contains the application server:
   a. In the administrative console, click **Servers > Application Servers.**
   b. Click the application server that you want to start automatically.
   c. Under Server Infrastructure, click **Java Process Management > Monitoring Policy** and change the setting of the Node Restart property to RUNNING.
   d. Click **Apply** and **Save** to save the configuration changes.
3. Add an autostart job entry to the QWAS61/QWAS61 subsystem. Enter the following command from the CL command line:

   ```
   ADDAJE SBSD(QWAS61/QWAS61) JOB(myserv) JOBD(mywasjobd/myserv)
   ```
4. **Optional:** Configure the system such that the QWAS61 subsystem starts during system startup. To enable automatic startup, add the following line to the system startup program:

   ```
   STRSBS QWAS61/QWAS61
   ```

   Notes:
   - The system startup program is defined by the QSTRUPPGM system value.
   - TCP/IP must be active before WebSphere Application Server subsystem can start. Ensure that the STRTCP command runs before the STRSBS QWAS61/QWAS61 command in your startup program or in your autostart job.
   - For more information about the QSTRUPPGM system value, see the *Work Management Guide*, SC41-5306, which is available at:

     `http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/QB3ALG03/CCONTENTS`

Shutting down the WebSphere Application Server subsystem

You can shutdown the WebSphere Application Server subsystem in order to completely shutdown the WebSphere Application Server environment.

For information about stopping individual servers, nodes and deployment managers, see “Starting an application server” on page 221 and “Stopping an application server” on page 226. In all cases, it is important to end the job gracefully so that the job can finish any tasks that are currently in progress, clean up any open connections, and end multiple threads in an appropriate order.

To stop (end) the WebSphere Application Server environment, perform one of the following steps. In both cases you must specify a value for the variable representing an adequate number of seconds.

To determine what is an adequate number of seconds, end the subsystem in a controlled fashion with DELAY(*NOLIMIT). The job logs for all WebSphere Application Server servers running in subsystem QWAS61 include message WAS0107 indicating that the server running in the job has ended. If the job is ended due to a SIGTERM signal (the signal is issued when commands such as ENDJOB, ENDSBS, etc are issued that result in the job being ended), the message contains the number of seconds required to gracefully end the application server. However, if the application server does not have enough time to end the application server gracefully, no message is issued.

If you must use the ENDSBS OPTION(*IMMED) command or ENDJOB OPTION(*IMMED), it is recommended that you set the amount of time available for handling the job termination signal to an appropriate value.

If no message is found in the job log, it is a good indication that you need to increase the amount of time specified on the ENDSBS or ENDJOB command.

1. Invoke the End Subsystem (ENDSBS) CL command specifying the QWAS61 subsystem.

   ```
   ENDSBS SBS(QWAS61) OPTION(*CNTRLD) DELAY(adequateNumberOfSeconds)
   ```

   where *adequateNumberOfSeconds* is appropriate for the amount of time it takes to stop the WebSphere Application Server servers running on your system.
2. Invoke the End Job (ENDJOB) CL command against the jobs running in the QWAS61 subsystem:

   ```
   ENDJOB JOB(jobNumber/QEJBSVR/jobName)
        OPTION(*CNTRLD) DELAY(adequateNumberOfSeconds)
   ```

   where *adequateNumberOfSeconds* is appropriate for the amount of time it takes to stop the WebSphere Application Server servers running on your system.

Creating application servers

WebSphere Application Server provides you with the capability to create application servers.

Determine if you want to use the application server you are creating as part of a cluster. If this application server is going to be part of a cluster, use the Create a new cluster wizard to create this application server.

You can use either the **createApplicationServer**, **createWebServer**, or the **createGenericServer** wsadmin commands (see the *Using the administrative clients* PDF), or the **Create New Application Server** wizard in the administrative console to create a new application server.

You can also create a new application server when you add a cluster member to a server cluster.

If you are migrating from a previous version of WebSphere Application Server, you can upgrade a portion of the nodes in a cell, while leaving others at the older product level. This means that, for a period of time, you might be managing servers that are running at two different release levels in the same cell. However, when you create a new server definition, you must use a server configuration template, and that template must be created from a server instance that matches the version of the node on which you are creating a server.

There are no restrictions on what you can do with the servers running on the newer release level.

To create a new application server:

1. In the administrative console, click **Servers > Application servers > New**. The Create New Application Server wizard starts.

   You can also create the new application server using the wsadmin **createApplicationServer** command. For a description of how to use this command, see the *Using the administrative clients* PDF.
2. Configure 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. Click **Next**.
   d. Select a server template for the new server. You can use a default application server template for your new server or you can use the template that is optimized to perform well for development uses. The new application server inherits all of the configuration settings of the template server.

e. Click **Next** and then select Generate Unique HTTP Ports if you want unique ports generated for the application server. 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.

f. Click **Next**. Review the settings for the new server. If you want to change any of the settings, click **Previous** until you return to a page where you can change that setting. If you do not want to make any changes, click **Finish**

g. Click Review, select Synchronize changes with Nodes, and then click **Save** to save your changes.

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

This newly created application server is configured with many default settings that do not display when you run the Create New Application Server wizard. To view all of the configuration settings for this application server in the administrative console, click **Servers > Application servers** and then click on the name of this application server. If necessary, use this page to change the configuration settings for this server.

You can also Set the client.encoding.override JVM argument to UTF-8 if you need to use multiple language encoding support in the administrative console.

Configuring application servers for UCS Transformation Format

You can use the client.encoding.override=UTF-8 JVM argument to configure an application server for UCS Transformation Format. This format enables an application server to handle most character encodings, including specialized mathematical and technical symbols.

The client.encoding.override=UTF-8 argument is provided for backwards compatibility. You should only specify this argument if you require multiple language encoding support in the administrative console and there is no other way for you to set the request character encoding required to parse post and query strings.

Before configuring an application server for UCS Transformation Format, you should try to either:

- Explicitly set the ServletRequest Encoding inside of the JSP or Servlet that is receiving the POST and or query string data, which is the preferred J2EE solution, or
- Enable the autoRequestEncoding, option, which uses the client's browser settings to determine the appropriate character encoding. Older browsers might not support this option.

Important: If the client.encoding.override=UTF-8 JVM argument is specified, the autoRequestEncoding option does not work even if it is enabled. Therefore, when an application server receives a client request, it checks to see if the charset option is set on the content type header of the request:

1. If it is set, the application server uses the content type header for character encoding.
2. If it is not set, the application server uses the character encoding that is specified for the default.client.encoding system property.
3. If neither charset nor the default.client.encoding system property is set, the application server uses the ISO-8859-1 character set.

The application server never checks for an Accept-Language header. However, if the autoRequestEncoding option is working, the application server checks for an Accept-Language header before checking to see if a character encoding is specified for the default.client.encoding system property.

To configure an application server for UCS Transformation Format:

1. In the administrative console, click **Servers > Application servers** and select the server you want to enable for UCS Transformation Format.
2. Click **Java and Process Management > Process Definition > Java Virtual Machine**.
3. Specify `-Dclient.encoding.override=UTF-8` for **Generic JVM Arguments** and click **OK**. When this argument is specified, UCS Transformation Format is used instead of the character encoding that would be used if the autoRequestEncoding option was in effect.
4. Click **Save** to save your changes.
5. Restart the application server.

The application server uses UCS Transformation Format for encoding.

Creating server templates

A server template is used to define the configuration settings for a new application server. When you create a new application server, you either select the default server template or a template you previously created, that is based on another, already existing application server. The default template is used if you do not specify a different template when you create the server.

To create a server template.

1. In the administrative console, click **Servers > Application Servers**, and then click **Templates**.

   You can also create server templates using the **createServerTemplate** wsadmin command. For more information, see he *Using the administrative clients* PDF.
2. On the Server templates page, click **New**.
3. From the list of servers, select the server that you want to use to create the new template, and then click **OK**.
4. Enter the name of the new template and, optionally, a description of that template that distinguishes it from your other templates.
5. click **OK**.

Your new template is on the list of server templates that you can use to create a new application server.

Listing server templates

The step below describes how to list your server templates.

In the administrative console, click **Servers > Application Servers** , and then click **Templates**. A list of all of the existing templates displays.

You can also list server templates using the **listServerTemplates** wsadmin command. For more information, see the *Using the administrative clients* PDF.

Deleting server templates

The steps below describe how to delete a server template.

1. In the administrative console, click **Servers > Application Servers**, and then click **Templates**.

   You can also use the **deleteServerTemplate** wsadmin command to delete server templates. For more information, see the *Using the administrative clients* PDF.
2. Select the template you want to delete, and click **Delete**.

The template you chose is removed from the list and cannot be used to create a new application server.

Enabling user profiles to run application servers with Operations Navigator

With the exception of the default user profile (QEJBSVR), all user profiles that are used to run an application server must be enabled through the Operations Navigator for the WebSphere Application Server.

The user profile that you use to run the Operations Navigator must have *SECADM special authority to perform this task.

To enable a user profile to use the Operations Navigator to run application servers:

1. Open the Operations Navigator.
2. Right click the icon for the WebSphere Application Server.
3. Select **Application Administration** on the pop-up menu.
4. Click the **Host Applications** tab.
5. Expand **QIBM_EJB_PRODUCT**.
6. Expand **QIBM_EJB_GROUP_OF_FUNCS**.
7. Select **QIBM_EJB_SERVER_FUNC**, and click **Customize**
8. In the Users and groups list box, select the user profile under which you want the application servers to run.
9. Click the top **Add** button to add the user profile to the Usage allowed list and then click **OK**.
10. On the **Application Administration** panel, click **OK**.

The selected user profile can use the Operations Navigator to run application servers.

Managing application servers

You can use the administrative console or command line tools to manage your application servers.

You can use either the administrative console or command line tools to manage your application servers.

Important: If you are migrating from a previous version of WebSphere Application Server, you can upgrade a portion of the nodes in a cell, while leaving others at the older release level. This means that, for a period of time, you might be managing servers that are running at different release levels in the same cell. In this mixed environment, there are some restrictions on what you can do with servers that are running the older release level. There are no restrictions on what you can do with the servers that are running on the newer release level. For details, see “Creating application servers” on page 208 and “Creating clusters” on page 309.

To use the administrative console to view and manage an application server:

To use the administrative console to view and manage an application server:

1. In the administrative console click **Servers > Application servers**. The Application servers page lists the application servers in your environment and the status of each of these servers. You can use this page to change the status of a listed application server.
2. Click the name of a listed application server to view or change the configuration settings for that application server.
3. Save any configuration changes you make.
4. Start an application server.
5. Monitor the running application servers.
6. Stop an application server.

7. Delete an application server.

   Tip: If the server you are deleting has applications or modules mapped to it, remap the modules to another server, or create a new server and remap the modules to the new server, before deleting the old server. After a server to which modules are mapped is deleted, the modules cannot be remapped to another server. Therefore, if you do not remap the modules to another server before deleting the old one, you must uninstall all of the modules that were mapped to the old server, and then reinstall them on a different server.

   a. In the administrative console, click **Servers > Application servers** to access the Application Servers page.
   b. Select an application server to delete.
   c. Click **Delete**.
   d. Click **OK** to confirm the deletion.

The application servers are properly configured and the appropriate application servers are running.

Create additional application server as required.

Server collection

Use this page to view information about and manage application servers, Java message service (JMS) servers, and Web servers. For the Network Deployment product, you can also use this page to view information about and manage generic servers.

Application servers

The Application servers page lists the application servers in the cell. You can use this page to start and stop these application servers.

If you are using the Network Deployment product and have created clusters, you can also use this page to manage application servers that are cluster members if the **Include cluster members in the collection** console page preference is selected. When this preference is selected, a Cluster Name column is included in the application server information table. If an application server is part of a cluster, the Cluster Name column specifies the name of that cluster. If this preference is not selected, the Cluster Name column does not appear in the table and application servers that are cluster members are not listed in the list of application servers and cannot be managed from this page.

If you are using the Network Deployment product, this page also displays the status of the application servers. The status indicates whether a server is running, stopped, or encountering problems. You can also use this page to create new application servers, create application server templates, or delete existing application servers.

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

Generic servers

The Generic servers page is only available for the Network Deployment product. This page lists the generic servers in the cell and displays the status of these generic servers. The status indicates whether a server is running, stopped, or encountering problems. You can use this page to start and terminate these generic servers. You can also use this page to create new generic servers, create generic server templates, or delete existing generic servers. However, the Stop and Stop Immediate buttons on the administrative console do not work for generic servers.

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

Java message service (JMS) servers

The JMS servers page lists the JMS servers in the cell. You can use this page to start and stop these JMS servers. Each JMS server provides the functions of the JMS provider for a node in your administrative domain. There can be at most one JMS server on each node in the administration domain, and any application server within the domain can access JMS resources served by any JMS server on any node in the domain. To view this administrative console page, click **Servers > JMS servers**.

Note: JMS servers apply only to WebSphere Application Server Version 5.x nodes. You cannot create a JMS server on a node that is running WebSphere Application Server 6.x, but the existing Version 5.x JMS servers continue to be displayed, and you can modify their properties. However, you cannot use this page to delete a Version 5.x JMS server.

Web servers

The Web servers page lists the Web servers in your administrative domain. You can use this page to generate and propagate a Web server plug-in configuration file, create new Web servers, create new Web server templates, or delete existing Web servers. You can also use this page to start and stop these Web servers. To view this administrative console page, click **Servers > Web servers**.

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

Node
Specifies the name of the node holding the server.

Version
Specifies the version of the WebSphere Application Server product on which the server runs.

Status
Indicates whether the server is started or stopped. (Network Deployment only)

Note that if the status is *Unavailable*, the node agent is not running in that node and you must restart the node agent before you can start the server.

Application server settings
Use this page to view or change the settings of an application server instance. An application server is a server which provides services required to run enterprise applications.

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

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information. The **Runtime** tab is available only when the server is running.

Name:

Specifies a logical name for the server. Server names must be unique within a node. However, for multiple nodes within a cluster, you may have different servers with the same server name as long as the server and node pair are unique. You cannot change the value that displays in this field.

For example, a server named *server1* in a node named *node1* in the same cluster with a server named *server1* in a node named *node2* is allowed. Configuring two servers named *server1* in the same node is not allowed. WebSphere Application Server uses the server name for administrative actions, such as referencing the server in scripting.

Data type	String
Default	server1

Run in development mode:

Enabling this option may reduce the startup time of an application server. This may include JVM settings such as disabling bytecode verification and reducing JIT compilation costs. Do not enable this setting on production servers. This setting is only available on application servers running WebSphere Application Server Version 6.0 and later.

This option is not supported in an i5/OS environment.

Specifies that you want to use the JVM settings **-Xverify** and **-Xquickstart** on startup. After selecting this option, save the configuration and restart the server to activate development mode.

The default setting for this option is `false`, which indicates that the server will not be started in development mode. Setting this option to `true` specifies that the server will be started in development mode (with settings that will speed server startup time).

Data type	Boolean
Default	false

Parallel start:

Select this field to start the server on multiple threads. This might shorten the startup time.

Specifies that you want the server components, services, and applications to start in parallel rather than sequentially.

The default setting for this option is `true`, which indicates that the server be started using multiple threads. Setting this option to `false` specifies that the server will not be started in using multiple threads (which may lengthen startup time).

Note that the order in which the applications start depends on the weights you assigned to each them. Applications that have the same weight are started in parallel. You set an application's weight with the *Starting weight* option on the **Applications > Enterprise Applications >** *application_name* page of the Administrative Console.

Data type	Boolean
Default	true

Class loader policy:

Select whether there is a single class loader to load all applications or a different class loader for each application.

Class loading mode:

Specifies whether the class loader should search in the parent class loader or in the application class loader first to load a class. The standard for Developer Kit class loaders and WebSphere Application Server class loaders is `Parent first`.

This field only applies if you set the Class loader policy field to `single`.

If you select `Parent last`, your application can override classes contained in the parent class loader, but this action can potentially result in ClassCastException or linkage errors if you have mixed use of overridden classes and non-overridden classes.

Access to internal server classes:

Specifies whether the applications can access many of the server implementation classes.

If you select `Allow`, applications can access many of the server implementation classes. If you select `Restrict`, applications can not access many of the server implementation classes. The applications get a ClassNotFoundException if they attempt to access those classes.

Applications typically use the supported APIs and do not need to access system internals.

Process Id:

The native operating system's process ID for this server.

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

Cell name:

The name of the cell in which this server is running.

The Cell name property is read only.

Node name:

The name of the node in which this server is running.

The Node name property is read only.

State:

The run-time execution state for this server.

The State property is read only.

Ports collection:

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

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

This page displays only when you are working with ports for application servers.

Port Name:

Specifies the name of a port. Each name must be unique within the server.

Host:

Specifies the IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a resource (such as the naming service, or administrative service).

Port:

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

Transport Details:

Provides a link to the transport chains associated with this port. If no transport chains are associated with this port, the string ″No associated transports″ appears in this column.

Ports settings:

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

For WebSphere Application Server for Network Deployment, you can view this administrative console page by clicking one of the following paths:
- **Servers > Application Servers >** *server_name* **> Ports >** *port_name*
- **Servers > JMS Servers >** *server_name* **> Security Port Endpoint**
- **Servers > JMS Servers >** *server_name* **> Ports >** *port_name*

Port Name:

Specifies the name of the port. The name must be unique within the server.

Note that this field displays only when you are defining a port for an application server. You can select either:

Well-known Port
When you select this option , you can select a previously defined port from the drop down list

User-defined Port
When you select this option, you must create a port with a new name by entering the name of the new port in the text box

Data type String

Host:

Specifies the IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a resource (such as the naming service, administrative service, or JMS broker).

For example, if the host name is `myhost`, the fully qualified DNS name can be `myhost.myco.com` and the IP address can be `155.123.88.201`.

Host names on the ports can be resolvable names or IP addresses. The server will bind to the specific host name or IP address that is supplied. That port will only be accessible through the IP address that is resolved from the given host name or IP address. The IP address may be of the IPv4 (Internet Protocol Version 4) format for all platforms, and IPv6 (Internet Protocol Version 6) format on specific operating systems where the server supports IPv6.

Data type String
Default * (asterisk)

Port:

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

Port numbers in the server can be reused among multiple ports as long as they have host names that resolve to unique IP addresses and there is not a port with the same port number and a wildcard (*) host name. A port number is valid in the range of 0 and 65535. 0 specifies that the server should bind to any ephemeral port available.

Important: Port sharing cannot be created using the administrative console. If you need to share a port, you must use wsadmin commands to define that port. You must also make sure that the same discrimination weights are defined for all of the transport channels associated with that port.

Protocol channels only accept their own protocol. However, application channels usually accept anything that reaches them. Therefore, for application channels, such as WebContainer or Proxy, you should specify larger discrimination weights when sharing levels with protocol channels, such as HTTP or SSL. The one exception to this rule is if you have application channels that perform discrimination tests faster than the protocol channels. For example, a JFAP channel is faster at deciding on a request than the SSL protocol channel, and should go first for performance reasons. However, the WebContainer and Proxy channels must always be last because they accept everything that is handed to them.

Data type Integer
Default None

The following table lists server endpoints and their respective port ranges. In contrast to the z/OS platform, on a distributed platform or on the i5/OS platform, the ORB_LISTENER_ADDRESS and the BOOTSTRAP_ADDRESS endpoints must not specify the same port.

Endpoint (port)	Acceptable values for the port field
BOOTSTRAP_ADDRESS	1 - 65536
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	1 - 65535
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	1 - 65535
DCS_UNICAST_ADDRESS	1 - 65536
DRS_CLIENT_ADDRESS	1 - 65536
ORB_LISTENER_ADDRESS	0 - 65535 (If 0 is specified, the server starts on any available port and does not use the location service daemon)
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	1 - 65535
SIB_ENDPOINT_ADDRESS	1 - 65536
SIB_ENDPOINT_SECURE_ADDRESS	1 - 65536
SIB_MQ_ENDPOINT_ADDRESS	1 - 65536
SIB_MQ_ENDPOINT_SECURE_ ADDRESS	1 - 65536
SOAP_CONNECTOR_ ADDRESS	1 - 65536
WC_adminhost	1 - 65536
WC_adminhost_secure	1 - 65536
WC_defaulthost	1 - 65536
WC_defaulthost_secure	1 - 65536
ORB_SSL_LISTENER_ADDRESS	Not supported on the distributed and iSeries platforms

Custom property collection:

Use this page to view and manage arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties.

The administrative console contains several Custom Properties pages that work similarly. To view one of these administrative pages, click a **Custom Properties** link.

Name:

Specifies the name (or key) for the property.

Each property name must be unique. If the same name is used for multiple properties, the value specified for the first property that has that name is used.

Do not start your property names with `was.` because this prefix is reserved for properties that are predefined in the application server.

Value:

Specifies the value paired with the specified name.

Description:

Provides information about the name-value pair.

Custom property settings:

Use this page to configure arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available in the administrative console.

To view this administrative console page, click one of the following paths:
- For an application server, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Administration** > **Custom Properties** .
- For a JMS server, click **Servers** > **JMS Servers** > *server_name*. Then, under Server Infrastructure, click **Administration** > **Custom Properties** .
- For a deployment manager, click For a deployment manager, click **System administration > Deployment manager**, and then under Server Infrastructure, click > **Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties**.

You can then click **New** to, create a new custom property, click on the name of an existing property to change its settings, or click **Delete** to delete an existing property.

Name:

Specifies the name (or key) for the property.

Each property name must be unique. If the same name is used for multiple properties, the value specified for the first property that has that name is used.

Do not start your property names with `was.` because this prefix is reserved for properties that are predefined in WebSphere Application Server.

Data type String

Value:

Specifies the value paired with the specified name.

Data type String

Description:

Provides information about the name and value pair.

Data type String

Server component collection:

Use this page to view information about and manage server component types such as application servers, messaging servers, or name servers.

To view this administrative console page, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Administration > Server Components**.

Type:

Specifies the type of internal server.

Server component settings:

Use this page to view or configure a server component instance.

To view this administrative console, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Administration > Server Components >** *server_component_name*.

Name:

Specifies the name of the component.

Data type String

Initial State:

Specifies the desired state of the component when the server process starts. The options are: *Started* and *Stopped*. The default is *Started*.

Data type String
Default Started

Generic server settings

Use this page to view or change the settings of a generic server.

A generic server is a server that is managed in the WebSphere Application Server administrative domain, although it is not a server that is supplied by the WebSphere Application Server product. The generic server can be any server or process that is necessary to support the Application Server environment, including a Java server, a C or C++ server or process, or a Remote Method Invocation (RMI) server.

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

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information. The **Runtime** tab is available only when the server is running.

Name:

Specifies a logical name for the generic server.

Generic server names must be unique within a node. For multiple nodes within a cluster, you can have different generic servers with the same server name as long as the server and node pair are unique. For example, a server named server1 in a node named node1 in the same cluster with a server named server1 in a node named node2 is allowed. Configuring two servers named server1 in the same node is not allowed. WebSphere Application Server uses the server name for administrative actions, such as referencing the server in scripting.

It is highly recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular WebSphere Application Servers. This will enable you to quickly determine whether to use the Terminate or Stop button in the administrative console to stop a specific application server.

You must use the Terminate button to stop a generic application server.

Data type String
Default

Environment entries collection

Use this page to view and manage arbitrary name-value pairs of data, where the name is a environment entry key and the value is a string value that can be used to set internal system configuration environment entries.

To view this page, in the administrative console click **Proxy Servers** > *server_name*, and then under Server Infrastructure, click **Java and Process Management** > **Environment Entries**.

Name

Specifies the name (or key) for the environment entry. The name is a string that is used to set an internal system configuration environment entry.

Each environment entry name must be unique. If the same name is used for multiple environment entries, the value specified for the first environment entry that has that name is used.

Do not start your environment entry names with `was.` because this prefix is reserved for environment entries that are predefined for WebSphere Application Server.

Value

Specifies the value paired with the specified name.

Description

Provides information about the name-value pair.

Environment entries settings

Use this page to configure arbitrary name-value pairs of data, where the name is an environment entry key and the value is a string value that can be used to set internal system configuration environment entries. Defining a new environment entry enables you to configure a setting beyond that which is available in the administrative console.

To view this page, in the administrative console click **Proxy Servers >** *server_name*. Under Server Infrastructure, click **Java and Process Management > Environment Entries**. Then do one of the following:
- Click **New** to create a new environment entry.
- Click the name of an existing environment entry to change its settings,
- Select an existing environment entry and click **Delete** to delete that entry.

Name:

Specifies the name (or key) for the environment entry.

Each environment entry name must be unique. If the same name is used for multiple environment entries, the value specified for the first environment entry that has that name is used.

Do not start an environment entry name with `was.` because this prefix is reserved for environment entries that are predefined in WebSphere Application Server.

Data type String

Value:

Specifies the value paired with the specified name.

Data type String

Description:

Provides information about the name and value pair.

Data type String

Starting an application server

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

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

This procedure for starting a server also normally applies to restarting a server. The one exception might be if a server fails and you want the recovery functions to complete their processing prior to new work being started on that server. In this situation you must restart the server in recovery mode.

After you create a new application server definition, you can start, stop, or manage the new server using the administrative console, or you can use commands to perform these tasks for the new server.

After you start an Application Server, other processes might not immediately discover the running application server. Application servers are discovered by the node agent. However, node agents are discovered by the deployment manager. Even though node agents usually discover local application servers quickly, it might take a deployment manager up to 60 seconds to discover a node agent.

If you are using clusters, the **Initial State** property of the application server subcomponent (**Servers** > **Application servers** > *server_name* > **Administration** > **Server Components** > **Application Server**) is not intended to be used to control the state of individual servers in the cluster at the time the cluster is started. It is intended only as a way to control the state of the Application Server subcomponent of a

server. It is best to start and stop the individual servers of a cluster using the Server options of the administrative console or command line commands (**startServer** and **stopServer**).

There are several options for starting an application server:

- You can use the administrative console:
  1. In the administrative console, click **Servers > Application servers** to determine the node agent on which the application server you are starting resides.
  2. Click **System administration > Nodes** and make sure the node agent is running.

     If the node agent is not running, issue the **startNode** command and then issue the **startServer** command. Once a node agent completely stops running and remains stopped, you cannot remotely start the node agent from the Node Agents page. You must issue the **startNode** command to start the node agent on the node where it runs.

     See the *Using the administrative clients* PDF for more information on how to issue these commands.
  3. Click **Servers > Application servers** again and select the application server you want to start.
  4. Click **Start**. You can view the **Status** value and any messages or logs to make sure the application server starts.
- You can use the **startServer** Qshell command. The **startServer** Qshell command allows you to start a single application server.
- You can use the **Submit Job (SBMJOB) CL** command to start an application server. You can run the following CL command from an i5/OS command line.

  ```
  SBMJOB CMD(CALL PGM(QWAS61/QWASSTRSVR) PARM('-profilePath'
    'profile_root' '-server' 'server')) JOB(server)
    JOBD(QWAS61/QWASJOBD) JOBQ(QWAS61/QWASJOBQ) USER(QEJBSVR) LANGID(*USRPRF)
    CNTRYID(*USRPRF) CCSID(*USRPRF) OUTQ(QWAS61/QWASOUTQ)
  ```
- **Optional:** Start an application server with tracing and debugging active.

  To start the Application Server with standard Java debugging enabled:
  1. In the administrative console, click **Servers > Application Servers**, click the application server whose processes you want to trace and debug, and then click **Java and Process Management > Process Definition > Java Virtual Machine**.
  2. On the Java virtual machine page, select the **Debug Mode** setting to enable the standard Java debugger. If needed, set debug arguments. Then, click **OK**.
  3. Save the changes to a configuration file.
  4. Stop the Application Server.
  5. Start the Application Server again as previously described.

After the server starts, install your applications.

You can use one of these same options to stop an application server.

Restarting an application server in recovery mode

When an application server instance with active transactions in progress restarts after a failure, the transaction service uses recovery logs to complete the recovery process. These logs, which each transactional resource maintains, are used to rerun any InDoubt transactions and return the overall system to a self-consistent state.

When you restart an application server in recovery mode:

- Transactional resources complete the actions in their recovery logs and then shut down. This action frees up any resource locks that the application server held prior to the failure.
- During the recovery period, only the subset of application server functions that are necessary for transactional recovery to proceed are available.
- The application server does not accept new work during the recovery process.

- The application server shuts down when the recovery is complete.

This recovery process begins as soon as all of the necessary subsystems within the application server are available. If the application server is not restarted in recovery mode, the application server can start accepting new work as soon as the server is ready, which might occur before the recovery work has completed.

Normally, this process is not a problem. However, situations exist when your operating procedures might not be compatible with supporting recovery work and new work simultaneously. For example, you might have a high availability environment where the work handled by the application server that failed is immediately moved to another application server. This backup application server then exclusively processes the work from the application server that failed until recovery has completed on the failed application server and the two application servers can be re-synchronized. In this situation, you might want the failing application server to only perform its transactional recovery process and then shut down. You might not want this application server to start accepting new work while the recovery process is taking place.

To prevent the assignment of new work to an application server that is going through its transaction recovery process, restart the application server in recovery mode.

When you restart a failed application server, the node agent for the node on which the failed application server resides must be running before you can restart that application server.

If you want to be able restart an application server in recovery mode, you must perform the following steps before a failure occurs, and then restart the application server to enable your configuration changes:

- If the server is monitored by a node agent, you must clear the Automatic restart option for that server. Clearing this option prevents the node agent from automatically restarting the server in normal mode, before you have a chance to start it in recovery mode.
  1. In the administrative console, click **Servers > Application Servers >** *server_name*.
  2. Under Server Infrastructure, click **Java and Process Management > Monitoring Policy**.
  3. Clear the Automatic restart option.
- If a catastrophic failure occurs that leaves InDoubt transactions, issue the **startServer** *server_name* **-recovery** command from the command line. This command restarts the server in recovery mode. You must issue the command from the *install_root*/profiles/*profile_name*/bin directory for the profile with which the server is associated.

The application server restarts in recovery mode, performs transactional recovery, and shuts down. Any resource locks that the application server held prior to the failure are released.

Running application servers under specific user profiles

You can run an application server under a user profile other than the default QEJBSVR user profile.

To change the application server default user profile:

1. Choose an existing user profile, or use the Create User Profile (CRTUSRPRF) command to create a new user profile. The new user profile must have authority to the same objects to which the QEJBSVR user profile has authority.
   a. Specify QEJBSVR as the profile for the new profile group.
   b. Issue the following command from the command line:
      ```
      CHGUSRPRF USRPRF(profile) GRPPRF(QEJBSVR)
      ```

      Important: If you use the enbprfwas script to enable your user profile, you do not have to issue this command. Instead specify the -chggrpprf parameter on the enbprfwas script.

2. **Optional:** If the application server is currently running under a user profile other than QEJBSVR, run the following commands in Qshell:
   a. chown -R QEJBSVR *profile_root*
   b. *app_server_root*/bin/grtwasaut -profileName profile_name -user QEJBSVR -dtaaut RWX -objaut ALL -recursive
   c. **Optional:** If the old user profile is no longer used to run any of the servers in the instance, you can issue the following command to revoke that profile's authorities:

      ```
      app_server_root/bin/rvkwasaut
        -profileName profile_name
        -user profile -recursive
      ```

      where *profile_name* is the name of the old user profile. See the descriptions of the grtwasaut and rvkwasaut commands in the *Using the administrative clients* PDF for more information.
3. Perform one of the following actions to enable the user profile to run the application server:
   a. Use the Operations Navigator.
   b. Use the **enbprfwas** command For example, run the following command:

      ```
      app_server_root/bin/enbprfwas -profile myProfile
      ```

      where *myProfile* is the name of the user profile.
4. Specify the new user profile name in the application server Run As User property.
   a. In the administrative console, click **Servers > Application Servers** and select an application server.
   b. Under Server Infrastructure, select Java and Process Management, and click **Process Definition**.
   c. Under Additional Properties, click **Process Execution** and enter the name of the user profile in the Run As User field.
   d. Click **OK** and **Save** to save your configuration changes.
   a. Restart the application server.

You can now use the specified user profile to run application servers.

Running application servers from a non-root user

By default, the root user ID is used to run all application server processes on a Linux and UNIX platform. However, you can run all application server processes under the same non-root user and user group. This task describes how to run an application server process from a non-root user.

If administrative security is enabled, the user account repositories must not be the local operating system. Using the local operating system user registry requires the node agent to run as root. Refer to the *Securing applications and their environment* PDF for details.

If you are using the Tivoli Access Manager to perform authentication or authorization for WebSphere Application Server, it is important to be aware of potential permissions problems. For more information, see the *Securing applications and their environment* PDF.

When WebSphere Application Server is run as a UNIX user, it can only access files owned by its primary group. If it tries to access files by its secondary group, a java.io.FileNotFoundException: will occur because the file access permissions do not allow this type of access.

Run your application servers as non-root when you no longer want to use root authority. For security or administrative reasons, you may want to change to non-root user IDs. Perform this task at any time to change the permissions of an application server. You must restart the application server in order for the changes to take effect.

For the following steps, assume that:

- was1 is the user to run the application server
- wasgroup is the primary user group for user was1
- wasnode is the node name
- server1 is the application server
- /opt/WebSphere/AppServer is the installation root
- nodeProfile1 is the profile name.

For information about creating a profile, see manageprofiles command.

To configure an application server to run as non-root, complete the following steps.

1. Log on to the application server system as the root user.
2. Create the user ID was1 with a primary user group of wasgroup. The user ID, was1, is an example. You can name the user something else.
3. Log off and back on as root.
4. Start server1 as root. Run the startServer.sh script from the /bin directory of the installation root:
   ```
   startServer.sh server1
   ```
5. Specify user and group ID values for the **Run As User** and **Run As Group** settings for a server:
   a. Start the administrative console.
   b. Go to the Process execution page of the administrative console. You must define all three properties in the following table. In the administrative console, click **Servers > Application Servers >** *server*, and then under Server Infrastructure, click **Java and Process Management > Process Definition > Process Execution**, and change all of the following values:

Property	Value
Run As User	was1
Run As Group	wasgroup
UMASK	022 The value `022` means the files the process creates are writable by the group and by others as defined on the Linux or UNIX platforms.

   c. Click **OK**.
   d. Save the configuration.
6. Stop the application server. Use the stopServer.sh script from the /bin directory of the installation root:
   ```
   stopServer.sh server1
   ```
7. Change file permissions as the root user. The following example assumes that the installation root directory for WebSphere Application Server is /opt/WebSphere/AppServer:
   ```
   chgrp wasgroup /opt/WebSphere
   chgrp wasgroup /opt/WebSphere/AppServer
   chgrp -R wasgroup /opt/WebSphere/AppServer/profiles
   chgrp -R wasgroup /opt/WebSphere/AppServer/profiles/profile name
   chgrp -R wasgroup /opt/WebSphere/AppServer/logs
   chgrp -R wasgroup /opt/WebSphere/AppServer/properties
   chgrp -R wasgroup /opt/WebSphere/AppServer/temp
   chgrp -R wasgroup /opt/WebSphere/AppServer/cloudscape
   chgrp -R wasgroup /opt/WebSphere/AppServer/bin
   chgrp -R wasgroup /opt/WebSphere/AppServer/java
   chgrp -R wasgroup /opt/WebSphere/AppServer/lib
   chgrp -R wasgroup /opt/WebSphere/AppServer/installedChannels
   chgrp -R wasgroup /opt/WebSphere/AppServer/profiles/profile name/installedFilters
   chgrp -R wasgroup /opt/WebSphere/AppServer/etc/
   chgrp -R wasgroup /opt/WebSphere/AppServer/classes
   chgrp -R wasgroup /opt/WebSphere/AppServer/systemApps
   ```

```
chmod g+wr /opt/WebSphere
chmod g+wr /opt/WebSphere/AppServer
chmod -R g+wr /opt/WebSphere/AppServer/profiles
chmod -R g+rw /opt/WebSphere/AppServer/profiles/profile name
chmod -R g+wr /opt/WebSphere/AppServer/logs
chmod -R g+wr /opt/WebSphere/AppServer/properties
chmod -R g+wr /opt/WebSphere/AppServer/temp
chmod -R g+wr /opt/WebSphere/AppServer/cloudscape
chmod -R g+r /opt/WebSphere/AppServer/bin
chmod -R g+r /opt/WebSphere/AppServer/java
chmod -R g+r /opt/WebSphere/AppServer/lib
chmod -R g+rw /opt/WebSphere/AppServer/installedChannels
chmod -R g+rw /opt/WebSphere/AppServer/profiles/profile name/installedFilters
chmod -R g+rw /opt/WebSphere/AppServer/etc/
chmod -R g+rw /opt/WebSphere/AppServer/classes
chmod -R g+rw /opt/WebSphere/AppServer/systemApps
```

8. Log on to the application server system as user was1.
9. Start server1 as was1. Run the startServer.sh script from the /bin directory of the installation root:

   ```
   startServer.sh server1
   ```

10. If creating another server with a different user ID, follow this procedure again for the new user ID and server name.

    The two user IDs must share the same group, wasgroup.

You can start an application server from a non-root user.

Detecting and handling problems with runtime components

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

1. Regularly examine the status of runtime components.

   Browse messages displayed under WebSphere Runtime Messages in the status area at the bottom of the console. The runtime event messages, marked with a red X, provide detailed information on event processing.
2. If an application stops runningl, examine the status of the application If an application stops running when it should be operational, examine the status of the application on an Applications page and try restarting the application. If messages indicate that a server has stopped running, use the Application servers page to try restarting the server. If a cluster of servers stops running, use the Server Cluster page to try to restart 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 runtime components do not restart, reexamine the messages and read information on problem determination to help you to restart the components.

Stopping an application server

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

There are several options for stopping an application server:

- You can use the administrative console to stop an application server:
  1. In the administrative console, click **Servers > Application Servers**.
  2. Select the application server that you want stopped and click **Stop**.
  3. Confirm that you want to stop the application server.
  4. View the **Status** value and any messages or logs to see whether the application server stops.
- You can use the stopserver Qshell script to stop an application server:
- You can use the `End Job (ENDJOB) CL` command to stop an application server: To use the `ENDJOB CL` command, enter:

```
ENDJOB JOB(jobNumber/QEJBSVR/jobName) OPTION(*CNTRLD) DELAY(delayTime)
```

where *jobNumber* is the job number, *jobName* is the name of the application server job, and *delayTime* is the amount of time to wait for the job to end in seconds. It is recommended that delayTime be 600 seconds initially. To find out what the appropriate *delayTime* is, see the topic Stop the WebSphere Application Server environment in the i5/OS Information Center.

If you experience any problems shutting down a server, see the *Troubleshooting and support* PDF.

Core group service settings

Use this page to set up the application server properties that relate to core groups.

To view this administrative console page, click **Servers > Application servers>** > *server*. Then under Additional properties, select **Core group service**.

Click **Save** to save and synchronize your changes with all managed nodes.

Enable service at server startup

Select if you want the core group service, also known as the high availability manager service, to start on this process when the server starts. The core group service must be started before high availability functions, such as routing, and failover, work properly.

Important: Before disabling the core group service for a server process, see "When to use a high availability manager" on page 347 for a description of environments that might not require high availability functions.

Default Core group service starts when the server starts.

Core group name

Specifies the name of the core group that contains this application server as a member. To move a server to a different core group, in the administrative console, click **Servers > Core groups > Core group settings** > *core_group* > **Core group servers**.

Data type String

Allow activation

Select if high availability group members can be activated on this application server.

Is alive timer

Specifies the time interval, in seconds, at which the high availability manager will check the health of all of the active high availability group members that are running in this application server process. An active group member is a member that is able to accept work. If a group member fails, the application server on which the group member resides is restarted. If -1 is specified, the timer is disabled. If 0 (zero) is specified, the default value of 120 seconds is used.

Important: The value specified for this property can be overridden for the high availability groups using a particular policy if the Is alive timer property for that policy specifies a different time interval. If the Is alive timer setting specified for a policy is greater than 0 (zero), the high availability manager uses that time interval, instead of the one specified at this level, when determining how frequently it should check the health of a high availability group member using that particular policy.

Data type Any integer between -1 and 600, inclusive
Default 120 seconds

Transport buffer size
Specifies the buffer size, in megabytes, of the underlying group communication transport. The minimum buffer size is 10 megabytes.

Data type	String
Default	10 megabytes

Setting the time zone for a single application server
You can ensure that your application components use the correct time zone. How you do this varies by the operating system on which WebSphere Application Server is installed and, in some cases, by the scope required.

Verify that extended National Language Support (NLS) is installed on your i5/OS server. If extended NLS support is not already installed, install it by selecting option 21 when you install the base operating system (5769-SS1).

In some application environments, it is important that application server components use the same time zone.

You can change the time zone setting for a single application server, for all of the application servers running under a user profile, or for all of the JVM processes running on your i5/OS server.

- To change the time zone setting for a single application server, add the user.timezone Java virtual machine (JVM) custom property to your application server instance configuration settings.
- To change the time zone setting for all of the application servers running under a user profile, update the user.timezone property in the user profile properties file with the appropriate time zone setting.
- To change the time zone setting for all of the JVM processes running on that server, either update the user.timezone property in the properties file for your i5/OS server with the appropriate time zone setting or set a system local for that server.

To add the user.timezone Java virtual machine (JVM) custom property to the configuration settings of a specific application server:

1. In the administrative console click **Application Servers >** *server_name* .
2. Under Server Infrastructure, click **Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties**.
3. Specify `user.timezone` in the Name field and the appropriate time zone in the Value field.
4. **Optional:** Enter a description of the variable and the setting you specified.
5. Click **Apply**.
6. Save your work.

   Make sure Synchronize changes with Nodes is selected, and click **Save**.

All of the components of this application server use the time zone specified for the custom property.

Stop and restart this application server. You must restart the server for the change to take effect.

Setting the time zone for all of the application servers running under a user profile
You can update the user.timezone property in the properties file for a user profile to set the time zone for all of the application servers running under that user profile. Setting this property ensures that all application components running under that profile use the same time zone.

Verify that extended National Language Support (NLS) is installed on your i5/OS server. If extended NLS support is not already installed, install it by selecting option 21 when you install the base operating system (5769-SS1).

In some application environments, it is important that all of the application servers running under a user profile use the same time zone. If this is your situation, before starting your application servers, you can either update the user.timezone property in the SystemDefault.properties file for a specific user profile.

Important: The value specified for the user.timezone property in a user profile properties file overrides any system locale setting for the application servers running under that user profile.

1. Edit the SystemDefault.properties file located in the /home/*user_ID* directory. If the file does not exist, create a SystemDefault.properties file in that directory.
2. Change the value specified for the user.timezone property to the correct time zone. If this property does not exist, add it to the file.

   The syntax for setting the property is `user.timezone=`*`timezone`*, where *timezone* is the supported value for the appropriate time zone.

   The Java Virtual Machine (JVM) calculates the time based on the value of the user.timezone property and the QHOUR and QUTCOFFSET system values. QUTCOFFSET represents the number of hours' difference between the system's time zone and Greenwich Mean Time (GMT). The JVM adds the values of QHOUR and QUTCOFFSET to calculate GMT, and then uses GMT and value of the user.timezone property to derive the correct time.
3. Save your changes.

All of the components of the application servers running under this user profile use the time zone specified for the user.timezone property.

Stop and restart the application servers running under this user profile. You must restart these servers for the change to take effect.

Setting the same time zone for all of your JVM processes

You can set the same time zone for all of the JVM processes running on your i5/OS server.

Verify that extended National Language Support (NLS) is installed on your i5/OS server. If extended NLS support is not already installed, install it by selecting option 21 when you install the base operating system (5769-SS1).

In some application environments, it is important that all of your JVM processes use the same time zone. If this is your situation, before starting your application servers, you can either update the user.timezone property in the SystemDefault.properties file for your i5/OS server or configure a locale for that server:

1. Update the user.timezone property in the SystemDefault.properties file for your i5/OS server

   Important: The value you specify for the user.timezone property overrides any system locale setting you create.

   a. Edit the SystemDefault.properties file located in the /QIBM/UserData/Java400 directory. If the file does not exist, create a SystemDefault.properties file in that directory.

   b. Change the value specified for the user.timezone property to the correct time zone. If this property does not exist, add it to the file.

      The syntax for setting the property is `user.timezone=`*`timezone`*, where *timezone* is the supported value for the appropriate time zone.

      The Java virtual machine (JVM) calculates the time based on the value of the **user.timezone** property and the QHOUR and QUTCOFFSET system values. QUTCOFFSET represents the number of hours' difference between the system's time zone and Greenwich Mean Time (GMT). The JVM adds the values of QHOUR and QUTCOFFSET to calculate GMT, and then uses GMT and value of the user.timezone property to derive the correct time.

   c. Save your change.
2. Configure a system locale for your i5/OS server.

Important: If a value is specified for the user.timezone property in the SystemDefault.properties file, it overrides this system locale setting.

a. Create a locale source file.

   Run the Create File (CRTF) command to create this file from the LOCALSRC file in the QSYSLOCALE library.

b. Edit the source file by running the Start SEU (STRSEU) command.

c. Specify a time zone in the file.

   The source file also contains settings to indicate when daylight savings time begins, when it ends, and how much time to add or subtract. The Java virtual machine ignores these settings and reads only the TNAME time zone field. The value of TNAME must match the name of a Java time zone value.

d. Create the locale by running the Create Locale (CRTLOCALE) command.

e. Edit the user profile to use the new locale.

   To change the user profile under which the application server runs, run the Change User Profile (CHGUSRPRF) command.

f. Save your changes.

All of the JVM processes running on your i5/OS server use the same time zone.

Start your application servers.

Supported time zone values

Use this page as a reference for time zone variables that are supported by WebSphere Application Server.

The following table lists the time zone values that WebSphere Application Server supports:

- The **Time zone ID** column lists time zones, in boldface, and the locations within each time zone.
- The **Raw offset** column lists the difference, in hours and minutes, between Greenwich Mean Time (GMT) and the specified time zone.
- The **DST offset** column lists the offset, in minutes, for Daylight Savings Time (DST). If the field is blank, the time zone does not use DST.
- The **Display name** column lists the names of the time zones.
- The **QTIMZON variable** column only applies to the i5/OS operating system. The **QTIMZON variable** column lists the corresponding value for the QTIMZON system variable. If multiple values are specified in this column, either value is acceptable.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Etc/GMT+12	-12 : 00		GMT-12:00	
Etc/GMT+11	-11 : 00		GMT-11:00	
MIT	-11 : 00		West Samoa Time	
Pacific/Apia	-11 : 00		West Samoa Time	QN1100UTCS
Pacific/Midway	-11 : 00		Samoa Standard Time	
Pacific/Niue	-11 : 00		Niue Time	
Pacific/Pago_Pago	-11 : 00		Samoa Standard Time	
Pacific/Samoa	-11 : 00		Samoa Standard Time	
US/Samoa	-11 : 00		Samoa Standard Time	
America/Adak	-10 : 00	60	Hawaii-Aleutian Standard Time	QN1000HAST

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
America/Atka	-10 : 00	60	Hawaii-Aleutian Standard Time	
Etc/GMT+10	-10 : 00		GMT-10:00	
HST	-10 : 00		Hawaii Standard Time	
Pacific/Fakaofo	-10 : 00		Tokelau Time	
Pacific/Honolulu	-10 : 00		Hawaii Standard Time	QN1000UTCS
Pacific/Johnston	-10 : 00		Hawaii Standard Time	
Pacific/Rarotonga	-10 : 00		Cook Is. Time	
Pacific/Tahiti	-10 : 00		Tahiti Time	
SystemV/HST10	-10 : 00		Hawaii Standard Time	
US/Aleutian	-10 : 00	60	Hawaii-Aleutian Standard Time	
US/Hawaii	-10 : 00		Hawaii Standard Time	
Pacific/Marquesas	-9 : 30		Marquesas Time	
AST	-9 : 00	60	Alaska Standard Time	QN0900AST
America/Anchorage	-9 : 00	60	Alaska Standard Time	
America/Juneau	-9 : 00	60	Alaska Standard Time	
America/Nome	-9 : 00	60	Alaska Standard Time	
America/Yakutat	-9 : 00	60	Alaska Standard Time	
Etc/GMT+9	-9 : 00		GMT-09:00	
Pacific/Gambier	-9 : 00		Gambier Time	QN0900UTCS
SystemV/YST9	-9 : 00	60	Alaska Standard Time	
US/Alaska	-9 : 00	60	Alaska Standard Time	
America/Dawson	-8 : 00	60	Pacific Standard Time	
America/Ensenada	-8 : 00	60	Pacific Standard Time	
America/Los_Angeles	-8 : 00	60	Pacific Standard Time	
America/Tiajuana	-8 : 00	60	Pacific Standard Time	
America/Vancouver	-8 : 00	60	Pacific Standard Time	
America/Whitehorse	-8 : 00	60	Pacific Standard Time	
Canada/Pacific	-8 : 00	60	Pacific Standard Time	
Canada/Yukon	-8 : 00	60	Pacific Standard Time	
Etc/GMT+8	-8 : 00		GMT-08:00	
Mexico/BajaNorte	-8 : 00	60	Pacific Standard Time	
PST	-8 : 00	60	Pacific Standard Time	QN0800PST, QN0800U
PST8PDT	-8 : 00	60	Pacific Standard Time	
Pacific/Pitcairn	-8 : 00		Pitcairn Standard Time	QN0800UTCS
SystemV/PST8	-8 : 00		Pitcairn Standard Time	
SystemV/PST8PDT	-8 : 00	60	Pacific Standard Time	
US/Pacific	-8 : 00	60	Pacific Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
US/Pacific-New	-8 : 00	60	Pacific Standard Time	
America/Boise	-7 : 00	60	Mountain Standard Time	
America/Cambridge_Bay	-7 : 00	60	Mountain Standard Time	
America/Chihuahua	-7 : 00	60	Mountain Standard Time	
America/Dawson_Creek	-7 : 00		Mountain Standard Time	
America/Denver	-7 : 00	60	Mountain Standard Time	
America/Edmonton	-7 : 00	60	Mountain Standard Time	
America/Hermosillo	-7 : 00		Mountain Standard Time	
America/Inuvik	-7 : 00	60	Mountain Standard Time	
America/Mazatlan	-7 : 00	60	Mountain Standard Time	
America/Phoenix	-7 : 00		Mountain Standard Time	QN0700MST2, QN0700UTCS
America/Shiprock	-7 : 00	60	Mountain Standard Time	
America/Yellowknife	-7 : 00	60	Mountain Standard Time	
Canada/Mountain	-7 : 00	60	Mountain Standard Time	
Etc/GMT+7	-7 : 00		GMT-07:00	
MST	-7 : 00	60	Mountain Standard Time	QN0700MST, QN0700T
MST7MDT	-7 : 00	60	Mountain Standard Time	
Mexico/BajaSur	-7 : 00	60	Mountain Standard Time	
Navajo	-7 : 00	60	Mountain Standard Time	
PNT	-7 : 00	60	Mountain Standard Time	
SystemV/MST7	-7 : 00		Mountain Standard Time	
SystemV/MST7MDT	-7 : 00	60	Mountain Standard Time	
UA/Arizona	-7 : 00		Mountain Standard Time	
US/Mountain	-7 : 00	60	Mountain Standard Time	
America/Belize	-6 : 00		Central Standard Time	
America/Cancun	-6 : 00	60	Central Standard Time	
America/Chicago	-6 : 00	60	Central Standard Time	
America/Costa_Rica	-6 : 00		Central Standard Time	QN0600UTCS
America/El_Salvador	-6 : 00		Central Standard Time	
America/Guatemala	-6 : 00		Central Standard Time	
America/Managua	-6 : 00		Central Standard Time	
America/Menominee	-6 : 00	60	Central Standard Time	
America/Merida	-6 : 00	60	Central Standard Time	
America/Mexico_City	-6 : 00	60	Central Standard Time	
America/Monterrey	-6 : 00	60	Central Standard Time	
America/North_Dakota/Center	-6 : 00	60	Central Standard Time	
America/Rainy_River	-6 : 00	60	Central Standard Time	
America/Rankin_Inlet	-6 : 00	60	Central Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
America/Regina	-6 : 00		Central Standard Time	
America/Swift_Current	-6 : 00		Central Standard Time	
America/Tegucigalpa	-6 : 00		Central Standard Time	
America/Winnipeg	-6 : 00	60	Central Standard Time	
CST	-6 : 00	60	Central Standard Time	QN0600CST, QN600S
CST6CDT	-6 : 00	60	Central Standard Time	
Canada/Central	-6 : 00	60	Central Standard Time	
Canada/East-Saskatchewan	-6 : 00		Central Standard Time	
Canada/Saskatchewan	-6 : 00		Central Standard Time	
Chile/EasterIsland	-6 : 00	60	Easter Is.Time	
Etc/GMT+6	-6 : 00		GMT-06:00	
Mexico/General	-6 : 00	60	Central Standard Time	
Pacific/Easter	-6 : 00	60	Easter Is. Time	
Pacific/Galapagos	-6 : 00		Galapagos Time	
Pacific/Easter	-6 : 00	60	Easter Is. Time	
Pacific/Galapagos	-6 : 00		Galapagos Time	
SystemV/CST6	-6 : 00		Central Standard Time	
SystemV/CST6CDT	-6 : 00	60	Central Standard Time	
US/Central	-6 : 00	60	Central Standard Time	
America/Bogota	-5 : 00		Colombia Time	
America/Cayman	-5 : 00		Eastern Standard Time	
America/Detroit	-5 : 00	60	Eastern Standard Time	
America/Eirunepe	-5 : 00		Acre Time	
America/Fort_Wayne	-5 : 00		Eastern Standard Time	
America/Grand_Turk	-5 : 00	60	Eastern Standard Time	
America/Guayaquil	-5 : 00		Ecuador Time	
America/Havana	-5 : 00	60	Central Standard Time	
America/Indiana/Indianapolis	-5 : 00		Eastern Standard Time	
America/Indiana/Knox	-5 : 00		Eastern Standard Time	
America/Indiana/Marengo	-5 : 00		Eastern Standard Time	
America/Indiana/Vevay	-5 : 00		Eastern Standard Time	
America/Indianapolis	-5 : 00		Eastern Standard Time	QN0500UTCS
America/Iqaluit	-5 : 00	60	Eastern Standard Time	
America/Jamaica	-5 : 00		Eastern Standard Time	
America/Kentucky/Louisville	-5 : 00	60	Eastern Standard Time	
America/Kentucky/Monticello	-5 : 00	60	Eastern Standard Time	
America/Knox_IN	-5 : 00		Eastern Standard Time	
America/Lima	-5 : 00		Peru Time	
America/Louisville	-5 : 00	60	Eastern Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
America/Montreal	-5 : 00	60	Eastern Standard Time	
America/Nassau	-5 : 00	60	Eastern Standard Time	
America/New_York	-5 : 00	60	Eastern Standard Time	
America/Nipigon	-5 : 00	60	Eastern Standard Time	
America/Panama	-5 : 00		Eastern Standard Time	
America/Pangnirtung	-5 : 00	60	Eastern Standard Time	
America/Port-au-Prince	-5 : 00		Eastern Standard Time	
America/Porto_Acre	-5 : 00		Acre Time	
America/Rio_Branco	-5 : 00		Acre Time	
America/Thunder_Bay	-5 : 00	60	Eastern Standard Time	
Brazil/Acre	-5 : 00		Acre Time	
Canada/Eastern	-5 : 00	60	Eastern Standard Time	
Cuba	-5 : 00	60	Central Standard Time	
EST	-5 : 00	60	Eastern Standard Time	QN0500EST
EST5EDT	-5 : 00	60	Eastern Standard Time	
Etc/GMT+5	-5 : 00		GMT-05:00	
IET	-5 : 00		Eastern Standard Time	QN0500EST2
Jamaica	-5 : 00		Eastern Standard Time	
SystemV/EST5	-5 : 00		Eastern Standard Time	
SystemV/EST5EDT	-5 : 00	60	Eastern Standard Time	
US/East-Indiana	-5 : 00		Eastern Standard Time	
US/Eastern	-5 : 00	60	Eastern Standard Time	
US/Indiana-Starke	-5 : 00		Eastern Standard Time	
US/Michigan	-5 : 00	60	Eastern Standard Time	
America/Anguilla	-4 : 00		Atlantic Standard Time	
America/Antigua	-4 : 00		Atlantic Standard Time	
America/Aruba	-4 : 00		Atlantic Standard Time	
America/Asuncion	-4 : 00	60	Paraguay Time	
America/Barbados	-4 : 00		Atlantic Standard Time	
America/Boa_Vista	-4 : 00		Amazon Standard Time	
America/Caracas	-4 : 00		Venezuela Time	QN0400UTC2
America/Cuiaba	-4 : 00	60	Amazon Standard Time	
America/Curacao	-4 : 00		Atlantic Standard Time	
America/Dominica	-4 : 00		Atlantic Standard Time	
America/Glace_Bay	-4 : 00	60	Atlantic Standard Time	
America/Goose_Bay	-4 : 00	60	Atlantic Standard Time	
America/Grenada	-4 : 00		Atlantic Standard Time	
America/Guadeloupe	-4 : 00		Atlantic Standard Time	
America/Guyana	-4 : 00		Guyana Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
America/Halifax	-4 : 00	60	Atlantic Standard Time	
America/La_Paz	-4 : 00		Bolivia Time	
America/Manaus	-4 : 00		Amazon Standard Time	
America/Martinique	-4 : 00		Atlantic Standard Time	
America/Montserrat	-4 : 00		Atlantic Standard Time	
America/Port_of_Spain	-4 : 00		Atlantic Standard Time	
America/Porto_Velho	-4 : 00		Amazon Standard Time	
America/Puerto_Rico	-4 : 00		Atlantic Standard Time	QN0400UTCS
America/Santiago	-4 : 00	60	Chile Time	
America/Santo_Domingo	-4 : 00		Atlantic Standard Time	
America/St_Kitts	-4 : 00		Atlantic Standard Time	
America/St_Lucia	-4 : 00		Atlantic Standard Time	
America/St_Thomas	-4 : 00		Atlantic Standard Time	
America/St_Vincent	-4 : 00		Atlantic Standard Time	
America/Thule	-4 : 00	60	Atlantic Standard Time	
America/Tortola	-4 : 00		Atlantic Standard Time	
America/Virgin	-4 : 00		Atlantic Standard Time	
Antarctica/Palmer	-4 : 00	60	Chile Time	
Atlantic/Bermuda	-4 : 00	60	Atlantic Standard Time	QN0400AST
Atlantic/Stanley	-4 : 00	60	Falkland Is. Time	
Brazil/West	-4 : 00		Amazon Standard Time	
Canada/Atlantic	-4 : 00	60	Atlantic Standard Time	
Chile/Continental	-4 : 00	60	Chile Time	
Etc/GMT+4	-4 : 00		GMT-04:00	
PRT	-4 : 00		Atlantic Standard Time	
SystemV/AST4	-4 : 00		Atlantic Standard Time	
SystemV/AST4ADT	-4 : 00	60	Atlantic Standard Time	
America/St_Johns	-3 : 30	60	Newfoundland Standard Time	
CNT	-3 : 30	60	Newfoundland Standard Time	QN0330NST
Canada/Newfoundland	-3 : 30	60	Newfoundland Standard Time	
AGT	-3 : 00		Argentine Time	
America/Araguaina	-3 : 00	60	Brazil Time	
America/Belem	-3 : 00		Brazil Time	
America/Buenos_Aires	-3 : 00		Argentine Time	QN0300UTCS
America/Catamarca	-3 : 00		Argentine Time	
America/Cayenne	-3 : 00		French Guiana Time	
America/Cordoba	-3 : 00		Argentine Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
America/Fortaleza	-3 : 00		Brazil Time	
America/Godthab	-3 : 00	60	Western Greenland Time	
America/Jujuy	-3 : 00		Argentine Time	
America/Maceio	-3 : 00		Brazil Time	
America/Mendoza	-3 : 00		Argentine Time	
America/Miquelon	-3 : 00	60	Pierre & Miquelon Standard Time	
America/Montevideo	-3 : 00		Uruguay Time	
America/Paramaribo	-3 : 00		Suriname Time	
America/Recife	-3 : 00		Brazil Time	
America/Rosario	-3 : 00		Argentine Time	
America/Sao_Paulo	-3 : 00	60	Brazil Time	
Antarctica/Rothera	-3 : 00		Rothera Time	
BET	-3 : 00	60	Brazil Time	QN0300UTC2
Brazil/East	-3 : 00	60	Brazil Time	
Etc/GMT+3	-3 : 00		GMT-03:00	
America/Noronha	-2 : 00		Fernando de Noronha Time	QN0200UTCS
Atlantic/South_Georgia	-2 : 00		South Georgia Standard Time	
Brazil/DeNoronha	-2 : 00		Fernando de Noronha Time	
Etc/GMT+2	-2 : 00		GMT-02:00	
America/Scoresbysund	-1 : 00	60	Eastern Greenland Time	
Atlantic/Azores	-1 : 00	60	Azores Time	
Atlantic/Cape_Verde	-1 : 00		Cape Verde Time	QN0100UTCS
Etc/GMT+1	-1 : 00		GMT-01:00	
Africa/Abidjan	0 : 00		Greenwich Mean Time	
Africa/Accra	0 : 00		Greenwich Mean Time	
Africa/Bamako	0 : 00		Greenwich Mean Time	
Africa/Banjul	0 : 00		Greenwich Mean Time	
Africa/Bissau	0 : 00		Greenwich Mean Time	
Africa/Casablanca	0 : 00		Western European Time	
Africa/Conakry	0 : 00		Greenwich Mean Time	
Africa/Dakar	0 : 00		Greenwich Mean Time	
Africa/El_Aaiun	0 : 00		Western European Time	
Africa/Freetown	0 : 00		Greenwich Mean Time	
Africa/Lome	0 : 00		Greenwich Mean Time	
Africa/Monrovia	0 : 00		Greenwich Mean Time	
Africa/Nouakchott	0 : 00		Greenwich Mean Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Africa/Ouagadougou	0 : 00		Greenwich Mean Time	
Africa/Sao_Tome	0 : 00		Greenwich Mean Time	
Africa/Timbuktu	0 : 00		Greenwich Mean Time	
America/Danmarkshavn	0 : 00		Greenwich Mean Time	
Atlantic/Canary	0 : 00	60	Western European Time	
Atlantic/Faeroe	0 : 00	60	Western European Time	
Atlantic/Madeira	0 : 00	60	Western European Time	
Atlantic/Reykjavik	0 : 00		Greenwich Mean Time	
Atlantic/St_Helena	0 : 00		Greenwich Mean Time	
Eire	0 : 00	60	Greenwich Mean Time	
Etc/GMT	0 : 00		GMT+00:00	
Etc/GMT+0	0 : 00		GMT+00:00	
Etc/GMT-0	0 : 00		GMT+00:00	
Etc/GMT0	0 : 00		GMT+00:00	
Etc/Greenwich	0 : 00		Greenwich Mean Time	
Etc/UCT	0 : 00		Coordinated Universal Time	
Etc/UTC	0 : 00		Coordinated Universal Time	
Etc/Universal	0 : 00		Coordinated Universal Time	
Etc/Zulu	0 : 00		Coordinated Universal Time	
Europe/Belfast	0 : 00	60	Greenwich Mean Time	
Europe/Dublin	0 : 00	60	Greenwich Mean Time	
Europe/Lisbon	0 : 00	60	Western European Time	
Europe/London	0 : 00	60	Greenwich Mean Time	Q0000GMT2
GB	0 : 00	60	Greenwich Mean Time	
GB-Eire	0 : 00	60	Greenwich Mean Time	
GMT	0 : 00		Greenwich Mean Time	Q0000GMT
GMT0	0 : 00		GMT+00:00	
Greenwich	0 : 00		Greenwich Mean Time	
Iceland	0 : 00		Greenwich Mean Time	
Portugal	0 : 00	60	Western European Time	
UCT	0 : 00		Coordinated Universal Time	
UTC	0 : 00		Coordinated Universal Time	Q0000UTC
Universal	0 : 00		Coordinated Universal Time	
WET	0 : 00	60	Western European Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Zulu	0 : 00		Coordinated Universal Time	
Africa/Algiers	1 : 00		Central European Time	QP0100CET, QP0100UTCS
Africa/Bangui	1 : 00		Western African Time	
Africa/Brazzaville	1 : 00		Western African Time	
Africa/Ceuta	1 : 00	60	Central European Time	
Africa/Douala	1 : 00		Western African Time	
Africa/Kinshasa	1 : 00		Western African Time	
Africa/Lagos	1 : 00		Western African Time	
Africa/Libreville	1 : 00		Western African Time	
Africa/Luanda	1 : 00		Western African Time	
Africa/Malabo	1 : 00		Western African Time	
Africa/Ndjamena	1 : 00		Western African Time	
Africa/Niamey	1 : 00		Western African Time	
Africa/Porto-Novo	1 : 00		Western African Time	
Africa/Tunis	1 : 00		Central European Time	
Africa/Windhoek	1 : 00	60	Western African Time	
Arctic/Longyearbyen	1 : 00	60	Central European Time	
Atlantic/Jan_Mayen	1 : 00	60	Eastern Greenland Time	
CET	1 : 00	60	Central European Time	
ECT	1 : 00	60	Central European Time	QP0100CET3
Etc/GMT-1	1 : 00		GMT+01:00	
Europe/Amsterdam	1 : 00	60	Central European Time	
Europe/Andorra	1 : 00	60	Central European Time	
Europe/Belgrade	1 : 00	60	Central European Time	
Europe/Berlin	1 : 00	60	Central European Time	
Europe/Bratislava	1 : 00	60	Central European Time	
Europe/Brussels	1 : 00	60	Central European Time	
Europe/Budapest	1 : 00	60	Central European Time	
Europe/Copenhagen	1 : 00	60	Central European Time	
Europe/Gibraltar	1 : 00	60	Central European Time	
Europe/Ljubljana	1 : 00	60	Central European Time	
Europe/Luxembourg	1 : 00	60	Central European Time	
Europe/Madrid	1 : 00	60	Central European Time	
Europe/Malta	1 : 00	60	Central European Time	
Europe/Monaco	1 : 00	60	Central European Time	
Europe/Oslo	1 : 00	60	Central European Time	
Europe/Paris	1 : 00	60	Central European Time	
Europe/Prague	1 : 00	60	Central European Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Europe/Rome	1 : 00	60	Central European Time	
Europe/San_Marino	1 : 00	60	Central European Time	
Europe/Sarajevo	1 : 00	60	Central European Time	
Europe/Skopje	1 : 00	60	Central European Time	
Europe/Stockholm	1 : 00	60	Central European Time	
Europe/Tirane	1 : 00	60	Central European Time	
Europe/Vaduz	1 : 00	60	Central European Time	
Europe/Vatican	1 : 00	60	Central European Time	
Europe/Vienna	1 : 00	60	Central European Time	
Europe/Warsaw	1 : 00	60	Central European Time	
Europe/Zagreb	1 : 00	60	Central European Time	
Europe/Zurich	1 : 00	60	Central European Time	QP0100CET2
MET	1 : 00	60	Middle Europe Time	
Poland	1 : 00	60	Central European Time	
ART	2 : 00	60	Eastern European Time	
Africa/Blantyre	2 : 00		Central African Time	
Africa/Bujumbura	2 : 00		Central African Time	
Africa/Cairo	2 : 00	60	Eastern European Time	
Africa/Gaborone	2 : 00		Central African Time	
Africa/Harare	2 : 00		Central African Time	
Africa/Johannesburg	2 : 00		South Africa Standard Time	QP0200SAST
Africa/Kigali	2 : 00		Central African Time	
Africa/Lubumbashi	2 : 00		Central African Time	
Africa/Lusaka	2 : 00		Central African Time	
Africa/Maputo	2 : 00		Central African Time	
Africa/Maseru	2 : 00		South Africa Standard Time	
Africa/Mbabane	2 : 00		South Africa Standard Time	
Africa/Tripoli	2 : 00		Eastern European Time	
Asia/Amman	2 : 00	60	Eastern European Time	
Asia/Beirut	2 : 00	60	Eastern European Time	
Asia/Damascus	2 : 00	60	Eastern European Time	
Asia/Gaza	2 : 00	60	Eastern European Time	
Asia/Istanbul	2 : 00	60	Eastern European Time	
Asia/Jerusalem	2 : 00	60	Israel Standard Time	
Asia/Nicosia	2 : 00	60	Eastern European Time	
Asia/Tel_Aviv	2 : 00	60	Israel Standard Time	
CAT	2 : 00		Central African Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
EET	2 : 00	60	Eastern European Time	QP0200EET
Egypt	2 : 00	60	Eastern European Time	
Etc/GMT-2	2 : 00		GMT+02:00	
Europe/Athens	2 : 00	60	Eastern European Time	
Europe/Bucharest	2 : 00	60	Eastern European Time	
Europe/Chisinau	2 : 00	60	Eastern European Time	
Europe/Helsinki	2 : 00	60	Eastern European Time	
Europe/Istanbul	2 : 00	60	Eastern European Time	
Europe/Kaliningrad	2 : 00	60	Eastern European Time	
Europe/Kiev	2 : 00	60	Eastern European Time	
Europe/Minsk	2 : 00	60	Eastern European Time	
Europe/Nicosia	2 : 00	60	Eastern European Time	
Europe/Riga	2 : 00	60	Eastern European Time	
Europe/Simferopol	2 : 00	60	Eastern European Time	
Europe/Sofia	2 : 00	60	Eastern European Time	
Europe/Tallinn	2 : 00	60	Eastern European Time	QP0200EET2, QP0200UTCS
Europe/Tiraspol	2 : 00	60	Eastern European Time	
Europe/Uzhgorod	2 : 00	60	Eastern European Time	
Europe/Vilnius	2 : 00	60	Eastern European Time	
Europe/Zaporozhye	2 : 00	60	Eastern European Time	
Israel	2 : 00	60	Israel Standard Time	
Libya	2 : 00		Eastern European Time	
Turkey	2 : 00	60	Eastern European Time	
Africa/Addis_Ababa	3 : 00		Eastern African Time	QP0300UTCS
Africa/Asmera	3 : 00		Eastern African Time	
Africa/Dar_es_Salaam	3 : 00		Eastern African Time	
Africa/Djibouti	3 : 00		Eastern African Time	
Africa/Kampala	3 : 00		Eastern African Time	
Africa/Khartoum	3 : 00		Eastern African Time	
Africa/Mogadishu	3 : 00		Eastern African Time	
Africa/Nairobi	3 : 00		Eastern African Time	
Antarctica/Syowa	3 : 00		Syowa Time	
Asia/Aden	3 : 00		Arabia Standard Time	
Asia/Baghdad	3 : 00	60	Arabia Standard Time	
Asia/Bahrain	3 : 00		Arabia Standard Time	
Asia/Kuwait	3 : 00		Arabia Standard Time	
Asia/Qatar	3 : 00		Arabia Standard Time	
Asia/Riyadh	3 : 00		Arabia Standard Time	
EAT	3 : 00		Eastern African Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Etc/GMT-3	3 : 00		GMT+03:00	
Europe/Moscow	3 : 00	60	Moscow Standard Time	
Indian/Antananarivo	3 : 00		Eastern African Time	
Indian/Comoro	3 : 00		Eastern African Time	
Indian/Mayotte	3 : 00		Eastern African Time	
W-SU	3 : 00	60	Moscow Standard Time	
Asia/Riyadh87	3 : 07		GMT+03:07	
Asia/Riyadh88	3 : 07		GMT+03:07	
Asia/Riyadh89	3 : 07		GMT+03:07	
Mideast/Riyadh87	3 : 07		GMT+03:07	
Mideast/Riyadh88	3 : 07		GMT+03:07	
Mideast/Riyadh89	3 : 07		GMT+03:07	
Asia/Tehran	3 : 30	60	Iran Standard Time	
Iran	3 : 30	60	Iran Standard Time	
Asia/Aqtau	4 : 00	60	Aqtau Time	QP0400UTC2
Asia/Baku	4 : 00	60	Azerbaijan Time	
Asia/Dubai	4 : 00		Gulf Standard Time	QP0400UTCS
Asia/Muscat	4 : 00		Gulf Standard Time	
Asia/Oral	4 : 00	60	Oral Time	
Asia/Tbilisi	4 : 00	60	Georgia Time	
Asia/Yerevan	4 : 00	60	Armenia Time	
Etc/GMT-4	4 : 00		GMT+04:00	
Europe/Samara	4 : 00	60	Samara Time	
Indian/Mahe	4 : 00		Seychelles Time	
Indian/Mauritius	4 : 00		Mauritius Time	
Indian/Reunion	4 : 00		Reunion Time	
NET	4 : 00	60	Armenia Time	
Asia/Kabul	4 : 30		Afghanistan Time	
Asia/Aqtobe	5 : 00	60	Aqtobe Time	QP0500UTC2
Asia/Ashgabat	5 : 00		Turkmenistan Time	
Asia/Ashkhabad	5 : 00		Turkmenistan Time	
Asia/Bishkek	5 : 00	60	Kirgizstan Time	
Asia/Dushanbe	5 : 00		Tajikistan Time	
Asia/Karachi	5 : 00		Pakistan Time	QP0500UTCS
Asia/Samarkand	5 : 00		Turkmenistan Time	
Asia/Tashkent	5 : 00		Uzbekistan Time	
Asia/Yekaterinburg	5 : 00	60	Yekaterinburg Time	
Etc/GMT-5	5 : 00		GMT+05:00	
Indian/Kerguelen	5 : 00		French Southern & Antarctic Lands Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Indian/Maldives	5 : 00		Maldives Time	
PLT	5 : 00		Pakistan Time	
Asia/Calcutta	5 : 30		India Standard Time	
IST	5 : 30		India Standard Time	QP0530IST
Asia/Katmandu	5 : 45		Nepal Time	
Antarctica/Mawson	6 : 00		Mawson Time	
Antarctica/Vostok	6 : 00		Vostok Time	
Asia/Almaty	6 : 00	60	Alma-Ata Time	QP0600UTC2
Asia/Colombo	6 : 00		Sri Lanka Time	
Asia/Dacca	6 : 00		Bangladesh Time	
Asia/Dhaka	6 : 00		Bangladesh Time	QP0600UTCS
Asia/Novosibirsk	6 : 00	60	Novosibirsk Time	
Asia/Omsk	6 : 00	60	Omsk Time	
Asia/Qyzylorda	6 : 00	60	Qyzylorda Time	
Asia/Thimbu	6 : 00		Bhutan Time	
Asia/Thimphu	6 : 00		Bhutan Time	
BST	6 : 00		Bangladesh Time	
Etc/GMT-6	6 : 00		GMT+06:00	
Indian/Chagos	6 : 00		Indian Ocean Territory Time	
Asia/Rangoon	6 : 30		Myanmar Time	
Indian/Cocos	6 : 30		Cocos Islands Time	
Antarctica/Davis	7 : 00		Davis Time	
Asia/Bangkok	7 : 00		Indochina Time	
Asia/Hovd	7 : 00		Hovd Time	
Asia/Jakarta	7 : 00		West Indonesia Time	QP0700WIB
Asia/Krasnoyarsk	7 : 00	60	Krasnoyarsk Time	
Asia/Phnom_Penh	7 : 00		Indochina Time	
Asia/Pontianak	7 : 00		West Indonesia Time	
Asia/Saigon	7 : 00		Indochina Time	QP0700UTCS
Asia/Vientiane	7 : 00		Indochina Time	
Etc/GMT-7	7 : 00		GMT+07:00	
Indian/Christmas	7 : 00		Christmas Island Time	
VST	7 : 00		Indochina Time	
Antarctica/Casey	8 : 00		Western Standard Time (Australia)	
Asia/Brunei	8 : 00		Brunei Time	
Asia/Chongqing	8 : 00		China Standard Time	
Asia/Chungking	8 : 00		China Standard Time	
Asia/Harbin	8 : 00		China Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Asia/Hong_Kong	8 : 00		Hong Kong Time	QP0800JIST, QP0800UTCS
Asia/Irkutsk	8 : 00	60	Irkutsk Time	
Asia/Kashgar	8 : 00		China Standard Time	
Asia/Kuala_Lumpur	8 : 00		Malaysia Time	
Asia/Kuching	8 : 00		Malaysia Time	
Asia/Macao	8 : 00		China Standard Time	
Asia/Macau	8 : 00		China Standard Time	
Asia/Makassar	8 : 00		Central Indonesia Time	
Asia/Manila	8 : 00		Philippines Time	
Asia/Shanghai	8 : 00		China Standard Time	
Asia/Singapore	8 : 00		Singapore Time	
Asia/Taipei	8 : 00		China Standard Time	
Asia/Ujung_Pandang	8 : 00		Central Indonesia Time	QP0800WITA
Asia/Ulaanbaatar	8 : 00		Ulaanbaatar Time	
Asia/Ulan_Bator	8 : 00		Ulaanbaatar Time	
Asia/Urumqi	8 : 00		China Standard Time	
Australia/Perth	8 : 00		Western Standard Time (Australia)	QP0800AWST
Australia/West	8 : 00		Western Standard Time (Australia)	
CTT	8 : 00		China Standard Time	QP0800BST
Etc/GMT-8	8 : 00		GMT+08:00	
Hongkong	8 : 00		Hong Kong Time	
PRC	8 : 00		China Standard Time	
Singapore	8 : 00		Singapore Time	
Asia/Choibalsan	9 : 00		Choibalsan Time	
Asia/Dili	9 : 00		East Timor Time	
Asia/Jayapura	9 : 00		East Indonesia Time	QP0900WIT
Asia/Pyongyang	9 : 00		Korea Standard Time	
Asia/Seoul	9 : 00		Korea Standard Time	QP0900KST
Asia/Tokyo	9 : 00		Japan Standard Time	QP0900UTCS
Asia/Yakutsk	9 : 00	60	Yakutsk Time	
Etc/GMT-9	9 : 00		GMT+09:00	
JST	9 : 00		Japan Standard Time	QP0900JST
Japan	9 : 00		Japan Standard Time	
Pacific/Palau	9 : 00		Palau Time	
ROK	9 : 00		Korea Standard Time	
ACT	9 : 30		Central Standard Time (Northern Territory)	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Australia/Adelaide	9 : 30	60	Central Standard Time (South Australia)	QP0930ACST
Australia/Broken_Hill	9 : 30	60	Central Standard Time (South Australia/New South Wales)	
Australia/Darwin	9 : 30		Central Standard Time (Northern Territory)	
Australia/North	9 : 30		Central Standard Time (Northern Territory)	
Australia/South	9 : 30	60	Central Standard Time (South Australia)	
Australia/Yancowinna	9 : 30	60	Central Standard Time (South Australia/New South Wales)	
AET	10 : 00	60	Eastern Standard Time (New South Wales)	QP1000AEST
Antarctica/DumontDUrville	10 : 00		Dumont-d'Urville Time	
Asia/Sakhalin	10 : 00	60	Sakhalin Time	
Asia/Vladivostok	10 : 00	60	Vladivostok Time	
Australia/ACT	10 : 00	60	Eastern Standard Time (New South Wales)	
Australia/Brisbane	10 : 00		Eastern Standard Time (Queensland)	
Australia/Canberra	10 : 00	60	Eastern Standard Time (New South Wales)	
Australia/Hobart	10 : 00	60	Eastern Standard Time (Tasmania)	
Australia/Lindeman	10 : 00		Eastern Standard Time (Queensland)	
Australia/Melbourne	10 : 00	60	Eastern Standard Time (Victoria)	
Australia/NSW	10 : 00	60	Eastern Standard Time (New South Wales)	
Australia/Queensland	10 : 00		Eastern Standard Time (Queensland)	
Australia/Sydney	10 : 00	60	Eastern Standard Time (New South Wales)	
Australia/Tasmania	10 : 00	60	Eastern Standard Time (Tasmania)	
Australia/Victoria	10 : 00	60	Eastern Standard Time (Victoria)	
Etc/GMT-10	10 : 00		GMT+10:00	
Pacific/Guam	10 : 00		Chamorro Standard Time	QP1000UTCS
Pacific/Port_Moresby	10 : 00		Papua New Guinea Time	
Pacific/Saipan	10 : 00		Chamorro Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Pacific/Truk	10 : 00		Truk Time	
Pacific/Yap	10 : 00		Yap Time	
Australia/LHI	10 : 30	30	Load Howe Standard Time	
Australia/Lord_Howe	10 : 30	30	Load Howe Standard Time	
Asia/Magadan	11 : 00	60	Magadan Time	
Etc/GMT-11	11 : 00		GMT+11:00	
Pacific/Efate	11 : 00		Vanuatu Time	
Pacific/Guadalcanal	11 : 00		Solomon Is. Time	QP1100UTCS
Pacific/Kosrae	11 : 00		Kosrae Time	
Pacific/Noumea	11 : 00		New Caledonia Time	
Pacific/Ponape	11 : 00		Ponape Time	
SST	11 : 00		Solomon Is. Time	
Pacific/Norfolk	11 : 30		Norfolk Time	
Antarctica/McMurdo	12 : 00	60	New Zealand Standard Time	
Antarctica/South_Pole	12 : 00	60	New Zealand Standard Time	
Asia/Anadyr	12 : 00	60	Anadyr Time	
Asia/Kamchatka	12 : 00	60	Petropavlovsk-Kamchatski Time	
Etc/GMT-12	12 : 00		GMT+12:00	
Kwajalein	12 : 00		Marshall Islands Time	
NST	12 : 00	60	New Zealand Standard Time	QP1200NZST
NZ	12 : 00	60	New Zealand Standard Time	
Pacific/Auckland	12 : 00	60	New Zealand Standard Time	
Pacific/Fiji	12 : 00		Fiji Time	QN1200UTCS, QP1200UTCS
Pacific/Funafuti	12 : 00		Tuvalu Time	
Pacific/Kwajalein	12 : 00		Marshall Islands Time	
Pacific/Majuro	12 : 00		Marshall Islands Time	
Pacific/Nauru	12 : 00		Nauru Time	
Pacific/Tarawa	12 : 00		Gilbert Is. Time	
Pacific/Wake	12 : 00		Wake Time	
Pacific/Wallis	12 : 00		Wallis & Futuna Time	
NZ-CHAT	12 : 45	60	Chatham Standard Time	
Pacific/Chatham	12 : 45	60	Chatham Standard Time	QP1245UTCS
Etc/GMT-13	13 : 00		GMT+13:00	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Pacific/Enderbury	13 : 00		Phoenix Is. Time	
Pacific/Tongatapu	13 : 00		Tonga Time	
Etc/GMT-14	14 : 00		GMT+14:00	
Pacific/Kiritimati	14 : 00		Line Is. Time	

Changing the ports associated with an application server

You can use the administrative console or command line tools to manage your application servers.

Run the **chgwassvr** script command from the Qshell command line to change the ports for an application server.

1. On the I5/OS command line, issue the **STRQSH** command to start the Qshell. where *adequateNumberOfSeconds* is appropriate for the amount of time it takes to stop the WebSphere Application Server servers running on your system.
2. Run the **chgwassvr** script.

   See the *Using the administrative clients* PDF for more information about the **chgwassvr** script and the parameters you can specify.

   Examples:

   ```
   app_server_root/bin/chgwassvr -profileName devinst
    -server devsvr2 -portblock 11400
   ```

   In this example, the ports assigned to the application server devsvr2 in the profile devinst are changed.

   ```
   app_server_root/bin/chgwassvr -profileName devinst
    -server devsvr2 -admin 9093
   ```

   In this example, the administrative console port for the application server devsvr2 in the profile devinst is changed.

Web module or application server stops processing requests

Use this information to help determine why a Web module or application server has stopped processing new requests.

If an application server's process spontaneously closes, or its Web modules stop responding to new requests:

- Isolate the problem by installing Web modules on different servers, if possible.
- You can use the Tivoli performance viewer to determine which resources have reached their maximum capacity, such as Java heap memory (indicating a possible memory leak) and database connections. If a particular resource appears to have reached its maximum capacity, review the application code for a possible cause:
  - If database connections are used and never freed, ensure that application code performs a **close()** on any opened **Connection** object within a **finally{}** block.
  - If there is a steady increase in servlet engine threads in use, review application **synchronized** code blocks for possible deadlock conditions.
  - If there is a steady increase in a JVM heap size, review application code for memory leak opportunities, such as static (class-level) collections, that can cause objects to never get garbage-collected.

  See the *Tuning guide* PDF for more information about this tool.
- As an alternative to using the performance viewer to detect memory leak problems, enable verbose garbage collection on the application server. This feature adds detailed statements to the JVM error log file of the application server about the amount of available and in-use memory. To set up verbose garbage collection:

1. Select **Servers > Application Servers >** *server_name* **>Java and Process Management > Process Definition > Java Virtual Machine**, and enable **Verbose Garbage Collection**.
2. Stop and restart the application server.
3. Periodically, or after the application server stops, browse the log file for garbage collection statements. Look for statements beginning with ″allocation failure″. The string indicates that a need for memory allocation has triggered a JVM garbage collection (freeing of unused memory). Allocation failures themselves are normal and not necessarily indicative of a problem. The allocation failure statement is followed by statements showing how many bytes are needed and how many are allocated.

   If there is a steady increase in the total amount of free and used memory (the JVM keeps allocating more memory for itself), or if the JVM becomes unable to allocate as much memory as it needs (indicated by the bytes needed statement), there might be a memory leak.

- Force an application to create a thread dump (or javacore). Here is the process for forcing a thread dump, which is different from the process in earlier releases of the product:
  1. Using the wsadmin command prompt, get a handle to the problem application server:

     ```
     wsadmin>set jvm [$AdminControl completeObjectName type=JVM,process=server1,*]
     ```
  2. Generate the thread dump:

     ```
     wsadmin>$AdminControl invoke $jvm dumpThreads
     ```
  3. Look for an output file in the *profile_root*/logs directory with a name like javacore.*jobnum.jobuser.jobname.timestamp*.txt.
- Browse the thread dump for clues:
  – The thread dump shows information on the current Java heap size, and the minimum and maximum heap size settings. Look for an excessive current heap size.
  – The thread dump contains a snapshot of each thread in the process, starting in the section labeled ″Thread Information.″
    - Look for threads that are waiting on locks held by other threads.
    - Look for multiple threads in the same Java application code source location. Multiple threads from the same location might indicate a deadlock condition (multiple threads waiting on a monitor) or an infinite loop, and help identify the application code with the problem.

      It is normal for certain components in the WebSphere Application Server runtime to have certain types of threads in the same Java code source location. These components include the Web container, the EJB container and the ORB thread pool.

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

- http://www.ibm.com/servers/eserver/support/iseries/software/v5r3/index.html

Creating generic servers

A generic server is a server that is managed in the WebSphere administrative domain, although it is not a server that is supplied by WebSphere Application Server. The WebSphere Application Server generic servers function enables you to define a generic server as an application server instance within the WebSphere Application Server administration, and associate it with a non-WebSphere server or process.

There are two basic types of generic application servers:
- Non-Java applications or processes.
- Java applications or processes

Therefore, a generic server can be any server or process that is necessary to support the Application Server environment, including:
- A Java server
- A C or C++ server or process
- A CORBA server
- A Remote Method Invocation (RMI) server

You can use the wsadmin tool or the administrative console to create a generic server.

Important: For standalone application server profiles (profiles which do not belong to a Network Deployment cell), you can use the administrative console to create generic servers and to adjust the settings for the generic server. However, you cannot use the administrative console to start, stop or otherwise control the server. Use the wsadmin tool for those types of operations.

- **Create a non-Java application as a generic server.** The following steps describe how to use the administrative console to create a non-Java application as a generic application server.
  1. Select **Servers** > **Generic servers**
  2. Click **New**.
  3. Type in a name for the generic server.

     The name must be unique within the node. It is recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular WebSphere Application Servers.
  4. Select a template for the new server. You can use a default application server template for your new server or use an existing application server as a template. The new application server will inherit all properties of the template server. If you create the new server using an existing application server do not enable the option to map applications from the existing server to the new server. This option does not apply for a generic server.
  5. Click **Next**
  6. Click **Finish**. The generic server now appears as an option on the **Generic servers** page in the administrative console.
  7. On the **Generic servers** page, click on the name of the generic server.
  8. Under Additional Properties, click **Process Definition**.
  9. In the Executable name field under General Properties, enter the name of the non-WebSphere Application Server program that is launched when you start this generic server. Executable target type and Executable target properties are not used for non-Java applications. Executable target type and Executable target properties are only used for Java applications
  10. Click **OK**.
- **Create a Java application as a generic server:** The following steps describe how to use the administrative console to create a Java application as a generic application server.
  1. Select **Servers** > **Generic servers**
  2. Click **New**.
  3. Type in a name for the generic server.
  4. Click **Next**
  5. Click **Finish**. The generic server now appears as an option on the **Applications Server** page in the administrative console.
  6. Click **Finish**. The generic server now appears as an option on the **Generic servers** page in the administrative console.
  7. On the Generic servers page, click on the name of the generic server.
  8. Under Additional Properties, click **Process Definition**.
  9. In the Executable name field under General Properties, enter the path for the WebSphere Application Server default JVM, ${JAVA_HOME}/bin/java, which is used to run the Java application when you start this generic server.
  10. In the Executable target type field under General Properties, select whether a Java class name, **JAVA_CLASS**, or the name of an executable JAR file, **EXECUTABLE_JAR**, is used as the executable target of this Java process. The default for WebSphere Application Server is **JAVA_CLASS**.

11. In the Executable target field under General Properties, enter the name of the executable target. Depending on the executable target type, this is either a Java class containing a main() method, or the name of an executable JAR file.) The default for WebSphere Application Server is com.ibm.ws.runtime.WsServer.
12. Click **OK**.

    Note: If the generic server is to run an application server other than the WebSphere Application Server, leave the Executable name field set to the default value and specify the Java class containing the main function for your application serve in the Executable target field.

After you define a generic server, use the Application Server administrative console to start, stop, and monitor the associated non-WebSphere server or process when stopping or starting the applications that rely on them.

Important: You can use either the **Terminate** or **Stop** buttons in the administrative console to stop any application server, including a generic application server.

Starting and terminating generic application servers

This topic describes how to start and terminate generic servers.

If you create a generic server in a Network Deployment profile, you can use the administrative console to start and terminate this server.

1. Start a generic application server.

   There are two ways to start a generic server in a Network Deployment environment. You can use the MBean NodeAgent launchProcess operation of the wsadmin tool, or you can use the administrative console. To use the administrative console:

   a. In the administrative console, click **Servers > Application Servers**.
   b. Select the name of the generic server you want to start, and then click **Start**.
   a. View the **Status** value and any messages or logs to see whether the generic server starts.
2. Terminate generic servers.

   There are two ways to terminate a generic server in a Network Deployment environment. You can use the MBean terminate launchProcess operation of the wsadmin tool or you can use the administrative console. To use the administrative console:

   a. In the administrative console, click **Servers > Application Servers**.
   b. Select the check box beside the name of the generic server, and then click **Terminate**.

      Restriction: The **Stop** and **Stop Immediate** buttons on the administrative console do not work for generic servers.

   c. View the **Status** value and any messages or logs to see whether the generic server terminates.

Configuring transport chains

A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP or HTTP. Network ports can be shared among all of the channels within a chain. The channel framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing.

Ensure that a port is available for the new transport chain. If you need to set up a shared port, you must:

- Use wsadmin commands to create your transport chain.
- Make sure that all channels sharing that port have the same discrimination weight assigned to them.

You need to configure transport chains to provide networking services to such functions as the service integration bus component of IBM service integration technologies, WebSphere Secure Caching Proxy, and the high availability manager core group bridge service.

You can either use the administrative console or wsadmin commands to create a transport chain. If you want to use wsadmin commands, see the *Using the administrative clients* PDF for more information. If you use the administrative console, complete the following steps:

1. In the administrative console, click **Servers > Application servers >** *server_name*, and then select one of the following options, depending on the type of chain you are creating:
   - Under **SIP Container Settings**, click **SIP container transport chains**.
   - Under **Web container settings**, click **Web container transport chains**.
   - Under **Server messaging**, click either **Messaging engine inbound transports** or **WebSphere MQ link inbound transports**.
2. Click **New**. The Create New Transport Chain wizard initializes. During the transport chain creation process, you are asked to:
   - Specify a name for the new chain.
   - Select a transport chain template
   - Select a port, if one is available to which the new transport chain is bound. If a port is not available or you want to define a new port, specify a port name, the host name or IP address for that port, and a valid port number.

   When you click **Finish**, the new transport chain is added to the list of defined transport chains on the **Transport chain** panel.
3. Click the name of a transport chain to view the configuration settings that are in effect for the transport channels contained in that chain. To change any of these settings:
   a. Click the name of the channel whose settings you need to change.
   b. Change the configuration settings. Some of the settings, such as the port number are determined by what is specified for the transport chain when it is created and cannot be changed.
   c. Click on **Custom properties** to set any custom properties that are defined for your system.
4. When you your configuration changes, click **OK**.
5. Stop the application server and start it again.

   You must stop the application server and start it again before your changes take effect.

Update any routines you have that issue a call to start transports during server startup. When a routine issues a call to start transports during server startup, WebSphere Application Server converts the call to a transport channel call.

Transport chains

Transport chains represent a network protocol stack that is used for I/O operations within an application server environment. Transport chains are part of the channel framework function that provides a common networking service for all components, including the service integration bus component of IBM service integration technologies, WebSphere Secure Caching Proxy, and the high availability manager core group bridge service.

A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP, DCS. or HTTP. Network ports can be shared among all of the channels within a chain. The channel framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing.

transition: If you have a routine that issues a call to start transports during server startup, unless you have a mixed-node environment and that server is running in a V5.1 node, WebSphere Application Server converts the call to a transport chain call.

The transport chain configuration settings determine which I/O protocols are supported for that chain. Following are some of the more common types of channels. Custom channels that support requirements unique to a particular customer or environment can also be added to a transport chain.

DCS channel
Used by the core group bridge service, the data replication service (DRS), and the high availability manager to transfer data, objects, or events among application servers.

HTTP inbound channel
Used to enable communication with remote servers. It implements the HTTP 1.0 and 1.1 standards and is used by other channels, such as the Web container channel, to serve HTTP requests and to send HTTP specific information to servlets expecting this type of information.

HTTP inbound channels are used instead of HTTP transports to establish the request queue between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside.

HTTP proxy inbound channel
Used to handle HTTP requests between a proxy server and application server nodes.

HTTP Tunnel channel
Used to provide client applications with persistent HTTP connections to remote hosts that are either blocked by firewalls or require an HTTP proxy server (including authentication) or both. An HTTP Tunnel channel enables the exchange of application data in the body of an HTTP request or response that is sent to or received from a remote server. An HTTP Tunnel channel also enables client-side applications to poll the remote host and to use HTTP requests to either send data from the client or to receive data from an application server. In either case, neither the client nor the application server is aware that HTTP is being used to exchange the data.

JFAP channel
Used by the Java Message Service (JMS) server to create connections to JMS resources on a service integration bus.

MQ channel
Used in combination with other channels, such as a TCP channel, within the confines of WebSphere MQ support to facilitate communications between a WebSphere System Integration Bus and a WebSphere MQ client or queue manager.

SIP channel
Used to create a bridge in the transport chain between a session initiation protocol (SIP) inbound channel, and a servlet and JavaServer Page engine.

SIP container inbound channel
Used to handle communication between the SIP inbound channel and the SIP servlet container.

SIP inbound channel
Used to handle inbound SIP requests from a remote client.

SSL channel
Used to associate an Secure Sockets Layer (SSL) configuration repertoire with the transport chain. This channel is only available when SSL support is enabled for the transport chain. An SSL configuration repertoire is defined in the administrative console, under security, on the **SSL configuration repertoires > SSL configuration repertoires** page.

TCP channel
Used to provide client applications with persistent connections within a Local Area Network (LAN) when a node uses transmission control protocol (TCP) to retrieve information from a network.

UDP channel
Used to provide client applications with persistent connections within a Local Area Network (LAN) when a node uses user datagram protocol (UDP) to retrieve information from a network.

Web container channel
Used to create a bridge in the transport chain between an HTTP inbound channel and a servlet and JavaServer Page (JSP) engine.

HTTP transport collection

Use this page to view or manage HTTP transports. Transports provide request queues between WebSphere Application Server plug-ins for Web servers and Web containers in which the Web modules of applications reside. When you request an application in a Web browser, the request is passed to the Web server, then along the transport to the Web container.

Important: You can use HTTP transports only on a Version 5.1 node in a mixed WebSphere Application Server environment. You must use HTTP transport channels instead of HTTP transports to handle your HTTP requests on all of your other nodes.

The use of IPv6 (Internet Protocol Version 6) and WS-AT (Web Services Atomic Transactions) are not supported on HTTP transports; they are only supported on HTTP transport channel chains.

To view the HTTP Transport administrative console page, click **Servers > Application Servers >** *server_name* **> Web Container Settings > Web Container > HTTP Transports**.

Host

Specifies the host IP address to bind for transport. If the application server is on a local machine, the host name might be `localhost`.

Port

Specifies the port to bind for transport. The port number can be any port that currently is not in use on the system. The port number must be unique for each application server instance on a given machine.

For i5/OS and distributed platforms, there is no limit to the number of HTTP ports that are allowed per process.

SSL Enabled

Specifies whether to protect connections between the WebSphere plug-in and application server with Secure Sockets Layer (SSL). The default is not to use SSL.

HTTP transport settings

Use this page to view and configure an HTTP transport. The name of the page might be that of an SSL setting such as DefaultSSLSettings. This page will not be visible if you do not have an HTTP transport defined for your system.

Important: You can use HTTP transports only on a V5.1 node in a mixed WebSphere Application Server environment. You must use HTTP transport channels instead of HTTP transports to handle your HTTP requests on all of your other nodes.

The use of IPv6 (Internet Protocol Version 6) and WS-AT (Web Services Atomic Transactions) are not supported on HTTP transports; they are only supported on HTTP transport channel chains.

To view the HTTP Transport panel on the administrative console, click **Servers > Application Servers >** *server_name* **> Web Container Settings > Web Container > HTTP Transports >** *host_name*.

Host

Specifies the host IP address to bind for transport.

If the application server is on a local machine, the host name might be `localhost`.

Data type	String

Port

Specifies the port to bind for transport. Specify a port number between 1 and 65535. The port number must be unique for each application server on a given machine.

Data type	Integer
Range	1 to 65535

SSL Enabled

Specifies whether to protect connections between the WebSphere Application Server plug-in and application server with Secure Sockets Layer (SSL). The default is not to use SSL.

Data type	Boolean
Default	false

SSL

Specifies the Secure Sockets Layer (SSL) settings type for connections between the WebSphere Application Server plug-in and application server. The options include one or more SSL settings defined in the Security Center; for example, DefaultSSLSettings, ORBSSLSettings, or LDAPSSLSettings.

Data type	String
Default	An SSL setting defined in the Security Center

HTTP transport custom properties

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

Important: You can use HTTP transports only on a V5.1 node in a mixed WebSphere Application Server environment. You must use HTTP transport channels instead of HTTP transports to handle your HTTP requests on all of your other nodes.

The use of IPv6 (Internet Protocol Version 6) and WS-AT (Web Services Atomic Transactions) are not supported on HTTP transports; they are only supported on HTTP transport channel chains.

If you are using HTTP transports, you can set the following custom properties on either the Web Container or HTTP Transport Custom Properties page on the administrative console. When set on the Web container Custom Properties page, all transports inherit the properties. Setting the same properties on a transport overrides like settings defined for a Web container.

To specify custom properties for a specific transport on the HTTP Transport:

1. In the administrative console click **Servers > Application Servers >** *server_name* **> Web Container settings > Web Container >HTTP Transport**
2. Select a host.
3. Under **Additional Properties** select **Custom Properties**.
4. On the Custom Properties page, click **New**.
5. On the settings page, enter the property you want to configure in the **Name** field and the value you want to set it to in the **Value** field.
6. Click **Apply** or **OK**.
7. Click **Save** on the console task bar to save your configuration changes.
8. Restart the server.

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

ConnectionIOTimeOut:

Use the `ConnectionIOTimeOut` property to specify how long the J2EE server waits for an I/O operation to complete. Set this variable for each of the HTTP transport definitions on the server. You will need to set this variable for both SSL transport and non-SSL transport. Specifying a value of zero disables the time out function.

Data type	Integer
Default	For the i5/OS and distributed platforms: 5 seconds

ConnectionKeepAliveTimeout:

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

Data type	Integer
Default	For the i5/OS and distributed platforms: 5 seconds

MaxConnectBacklog: This property is only valid for i5/OS and distributed platforms. Use the `MaxConnectBacklog` property to specify the maximum number of outstanding connect requests that the operating system will buffer while it waits for the application server to accept the connections. If a client attempts to connect when this operating system buffer is full, the connect request will be rejected.

Keep in mind that a single client browser might need to open multiple concurrent connections (perhaps 4 or 5); however, also keep in mind that increasing this value consumes more kernel resources. The value of this property is specific to each transport.

Data type	Integer
Default	511

MaxKeepAliveConnections:

This property is only valid for i5/OS and distributed platforms. It is ignored on the z/OS platform because asynchronous I/O sockets are used to maintain connections in that environment.Use the `MaxKeepAliveConnections` property to specify the maximum number of concurrent keep alive (persistent) connections across all HTTP transports. To make a particular transport close connections after a request, you can set MaxKeepAliveConnections to `0` (zero) or you can set KeepAliveEnabled to `false` on that transport.

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

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

- The maximum number of requests for a connection was reached, preventing denial of service attacks in which a client tries to hold on to a keep-alive connection forever.
- A time out occurred while waiting to read the next request or to read the remainder of the current request.

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

MaxKeepAliveRequests:

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

On the i5/OS and distributed platforms, when this property is set to 0 (zero), the connection stays open as long as the application server is running.

Data type	Integer
Default	

KeepAliveEnabled: This property is only valid for i5/OS and distributed platforms. Use the `KeepAliveEnabled` property to specify whether or not to keep connections alive

Data type	String
Value	true or false
Default	true

RemoveServerHeader: Use this property to specify whether an existing server header is removed before a response message is sent. If this property is set to `true`, the value specified for the ServerHeaderValue property is ignored.

Data type	String
Value	true or false
Default	false

ServerHeaderValue: Use this property to specify a server header this is added to outgoing response messages if server header is not already provided. This property is ignored if the RemoveServerHeader property is set to true.

Data type	string
Default	WebSphere Application Server/*x.x* *x.x* is the version of WebSphere Application Server that you are using.

SoLingerValue: Use this property to specify, in seconds, the amount, that the socket close operation waits for data contained in the TCP/IP send buffer to be sent. This property is ignored if the UseSoLinger property is set to false.

Data type	Integer
Default	20 seconds

TcpNoDelay: Use this property to set the socket TCP_NODELAY option which enables and disables the use of the TCP Nagle algorithm for connections received on this transport. When this property is set to `true`, use of the Nagle algorithm is disabled.

Data type	String
Value	true or false
Default	true

Trusted: This property is only valid for i5/OS and distributed platforms. Use the `Trusted` property to indicate that the application server can use the private headers that the Web server plug-in adds to requests.

Data type	String
Value	true or false
Default	false

UseSoLinger: Use this property to set the socket SO_LINGER option. This property configures whether the socket close operation waits until all of the data contained in the TCP/IP send buffer is sent before closing a connection. If this property is enabled, and the time expires before the all of the content of the send buffer sent, any data remaining in the send buffer is lost.

The SoLingerValue property is ignored if this property is set to false.

Data type	String
Value	true or false
Default	true

HTTP transport channel custom properties

If you are using an HTTP transport channel, you can add the following custom property to the configuration settings for that HTTP transport channel.

To add a custom property:

1. In the administrative console, click **Application servers** > *server_name* **Web container settings** > **Web container transport chains** > *chain_name* > **HTTP Inbound Channel** > **Custom Properties** > **New**
2. Under **General Properties** specify the name of the custom property in the Name field and a value for this property in the Value field. You can also specify a description of this property in the Description field.
3. Click **Apply** or **OK**.
4. Click **Save** to save your configuration changes.
5. Restart the server.

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

inProcessLogFilenamePrefix

Use the inProcessLogFilenamePrefix property to specify a prefix for the filename of the network log file. Normally, when inprocess optimization is enabled, requests through the inprocess path are logged based

on the logging attributes set up for the Web container's network channel chain. You can use this property to add a prefix to the filename of the network log file. This new filename is then used as the filename for the log file for inprocess requests. Requests sent through the inprocess path are logged to this file instead of to the network log file. For example, if the log file for a network transport chain is named .../httpaccess.log, and this property is set to local for the HTTP channel in that chain, the filename of the log file for inprocess requests to the host associated with that chain is .../localhttpaccess.log.

Data type	String

listenBacklog

Use the listenBacklog property to specify the maximum number of outstanding connect requests that the operating system will buffer while it waits for the application server to accept the connections. If a client attempts to connect when this operating system buffer is full, the connect request will be rejected. Set this value to the number of concurrent connections that you would like to allow. Keep in mind that a single client browser might need to open multiple concurrent connection; also keep in mind that increasing this value consumes more kernel resources. The value of this property is specific to each transport.

Data type	Integer
Default	511

HTTP Tunnel transport channel custom property

If you are using an HTTP Tunnel transport channel, you can add the following custom property to the configuration settings for that HTTP Tunnel transport channel.

To add a custom property:

1. In the administrative console, click **Servers > Application servers >** *server_name* **> Ports**. Click on **View associated transports** for the HTTP Tunnel port to whose configuration settings you want to add this custom property.
2. Click **New**.
3. Under **General Properties** specify the name of the custom property in the Name field and a value for this property in the Value field. You can also specify a description of this property in the Description field.
4. Click **Apply** or **OK**.
5. Click **Save** to save your configuration changes.
6. Restart the server.

Following is a description of the custom property that is provided with the application server. This property is not shown on the settings page for an HTTP Tunnel transport channel.

pluginConfigurable

Indicates whether or not the configuration settings for the HTTP Tunnel transport channel are included in the plugin-cfg.xml file for the Web server associated with the application server that is using this channel. Configuration settings for each of the Web container transport channels defined for an application server are automatically included in the plugin-cfg.xml file for the Web server associated with that application server.

Data type	Boolean
Default	False

Web container transport custom properties

Use this page to set custom properties for a Web container transport.

Unless you are not migrating from an previous version of WebSphere Application Server, you must use HTTP transport channels instead of HTTP transports to handle your HTTP requests.

If you are using Web container transports, you can set the following custom properties on the Web Container **Custom Properties** panel on the administrative console. When set on the Web container **Custom Properties** page, all transports inherit the properties. Setting the same properties on a transport overrides like settings defined for a Web container.

To specify custom properties for a specific transport on the HTTP Transport:

1. In the administrative console click **Servers > Application Servers >** *server_name* **> Web Container settings > Web Container**
2. Select a host.
3. Under **Additional Properties** select **Custom Properties**.
4. On the Custom Properties page, click **New**.
5. On the settings page, enter the property you want to configure in the **Name** field and the value you want to set it to in the **Value** field.
6. Click **Apply** or **OK**.
7. Click **Save** on the console task bar to save your configuration changes.
8. Restart the server.

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

disableRequestMessageChunking

This custom property disables request message chunking when set to true. All the request body up to `maxRequestMessageBodySize` is buffered in memory.

Value

True or False. When a value is not specified, the default value is false.

maxRequestMessageBodySize

If `disableRequestMessageChunking` is false, this is the maximum amount of request body that is buffered in memory before sending the next chunk to the SR. If `disableRequestMessageChunking` is true, this is the maximum amount of request body data that is buffered in memory before sending the complete request to the SR. An HTTP 404 is returned if `maxRequestMessageBodySize` is exceeded.

Value

If `disableRequestMessageChunking` is false, the default size is 32K and maximum size is 10MB. If `disableRequestMessageChunking` is true, the default size is 10MB and the maximum size is 100MB.

Configuring inbound HTTP request chunking

Inbound HTTP request chunking, is configured at the Web container transport chain level. You can configure each Web container chain to enable or disable chunking. You can also configure the maximum message size for chunking disabled and the maximum chunk size for chunking enabled for each chain.

See the page for details on these settings.

The chains that host the Integrated Solutions console have chunking enabled by default, while all other Web container chains have chunking disabled by default. The reason for this is that there are some limitations in the use of inbound HTTP chunking on WebSphere Application Server for z/OS V6.1.

The following are limitations of inbound HTTP chunking

- The applications that are served by chains with chunking enabled must support chunked HTTP encoding. See RFC 2616 for more information on chunked encoding. If your application does not support chunked encoding, then you must map it to a Web container chain with chunking disabled.
- The current implementation requires a Web application to read the entire request, both headers and body, before sending any response data back to the client. If the Web application does not read the entire request, this results in an error in the servlet as well as an HTTP 500 Internal Server Error returned to the client.

1. In the administrative console click **Servers > Application Servers >** *server_name* **> Web Container settings > Web Container**
2. Select a host.
3. **Optional:** Enable request message chunking. See the article, "Web container transport custom properties" on page 257 for details on these settings.
   a. Under **Additional Properties** select **Custom Properties**.
   b. On the Custom Properties page, click **New**.
   c. On the settings page, enter the property, **disableRequestMessageChunking** in the **Name** field and the enter the value false in the **Value** field.
   d. Specify the maximum amount of request body that is buffered in memory before sending the next chunk to the SR.
   e. Click **Apply** or **OK**.
4. **Optional:** Disable request message chunking. See the article, "Web container transport custom properties" on page 257 for details on these settings.
   a. Under **Additional Properties** select **Custom Properties**.
   b. On the Custom Properties page, click **New**.
   c. On the settings page, enter the property, **disableRequestMessageChunking** in the **Name** field and the enter the value true in the **Value** field.
   d. Specify the maximum amount of request body data that is buffered in memory before sending the complete request to the SR.
   e. Click **Apply** or **OK**.
5. Click **Save** on the console task bar to save your configuration changes.
6. Restart the server.

Transport chain problems

Review the following topics if you encounter a transport chain problem.

TCP transport channel fails to bind to a specific host/port combination

If a TCP transport channel fails to bind to a specific port, one of the following situations might have occurred:

- You are trying to bind the channel to a port that is already bound to another application, such as another instance of a WebSphere Application Server.
- You are trying to bind to a port that is in a transitional state waiting for closure. This socket must transition to closed before you restart the server. The port might be in TIME_WAIT, FIN_WAIT_2, or CLOSE_WAIT state. Issue the `NETSTAT *CNN` command from a command prompt line to display the state of the port to which you are trying to bind.

  If you need to change the amount of elapse time that must occur before TCP/IP can release a closed connection and reuse its resources, see the *Tuning guide* PDF.

Deleting a transport chain

Transport chains cannot be deleted the same way that HTTP transports can be deleted. Because you cannot have multiple HTTP transports associated with the same port, when you delete an HTTP transport,

you effectively delete the associated port and stop all traffic on that port. However, the process is more complicated for a transport chain because multiple transport chains might be associated with the same port and you do not want to disrupt traffic on transport chains that you are not deleting.

Determine whether you want to delete a particular transport chain or all of the transport chains that are associated with a specific port.

You might have to delete one or more transport chains if you have to delete a port.

To delete a transport chain:

1. In the administrative console, click **Servers > Application servers >** *server* **> Ports**.
2. In the list of available ports, locate the port that you want to delete and click **View associated transports** for that port.
3. Select the transport chain you want to delete, and click **Delete**. If you intend to delete the port that is associated with this transport chain, repeat this step for all of the transport chains associated with this port.
4. Click **Save** to save your changes.

If you delete all of the transport chains associated with a port, you can delete the port.

Disabling ports and their associated transport chains

Transport chains cannot be disabled the same way that HTTP transports can be disabled. Because you cannot have multiple HTTP transports associated with the same port, when you disable an HTTP transport, you effectively disable the associated port and stop all traffic on that port. However, the process is more complicated for a port that has associated transport chains because multiple transport chains might be associated with the same port, and you might not want to disrupt traffic on all of the transport chains at the same time.

Determine whether you want to disable a particular transport chain or all of the transport chains that are associated with a specific port.

You might need to disable a transport chain if you want to temporarily stop all incoming traffic on a particular port or on a particular transport chain that is associated with that port.

To disable a specific transport chain:

1. In the administrative console, click **Servers > Application servers >** *server* **> Ports**.
2. In the list of available ports, locate the port that you want to delete and click **View associated transports** for that port.
3. Click the transport chain you want to disable.
4. Unselect the **Enabled** field, and click **OK**. If you want to temporarily stop all of the incoming traffic on a port, repeat this step for all of the transport chains associated with this port.
5. Click **Save** to save your changes.

When you want traffic to resume on these disabled transport chains, repeat the preceding steps for all of the transport chains you disabled, and select the **Enabled** field.

Transport chains collection

Use this page to view or manage transport chains. Transport chains enable communication through transports, or protocol stacks, which are usually socket based.

A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP or HTTP. Network ports can be shared among all of the channels within a chain. The Channel Framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing.

The **Transport chains** page lists the transport chains defined for the selected application server. Transport chains represent network protocol stacks operating within this application server.

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Ports**. Click on **View associated transports** for the port whose transport chains you want to view.

Name
Specifies a unique identifier for the transport chain. The name must consist of alphanumeric or national language characters and can start with a number. The name must be unique within a WebSphere Application Server configuration. Click on the name of a transport chain to change its configuration settings.

Enabled
When set to true, indicates that the transport chain is activated at application server startup.

Host
Specifies the host IP address to bind for transport. If the application server is on a local machine, the host name might be localhost.

Port
Specifies the port to bind for transport. The port number can be any port that currently is not in use on the system, might be localhost or the wildcard character * (an asterisk). The port number must be unique for each application server instance on a given machine

SSL Enabled
When enabled, users are notified that there is a channel that enables Secure Sockets Layer (SSL) in the listed chain. When SSL is enabled, all traffic going through this transport is encrypted and digitally secured.

Transport chain settings
Use this page to view a list of the types of transport channels configured for the selected transport chain. A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP, HTTP, or DCS.

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Ports**. Click on **View associated transports** for the port whose transport chains you want view and then click on the name of a specific chain.

Name
Specifies the name of the selected transport chain.

You can edit this field to rename this transport chain. However, remember that the name must be unique within a WebSphere Application Server configuration.

Enabled
When checked, this transport chain is activated at application server startup.

Transport channels
Lists the transport channels configured for this transport chain and their configuration settings. To change a transport channel's configuration settings, click on the name of that transport channel.

HTTP tunnel transport channel settings

Use this page to view and configure an HTTP tunnel transport channels. Inbound connections sent through this channel are tunneled over HTTP, allowing intermediates to view this data as the body of an HTTP message instead of in its natural format. This type of channel is often used to circumvent firewalls with protocol restrictions.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** . Click on **View associated transports** for the port associated with the HTTP Tunnel transport channel whose settings you want to look at.

Transport channel name

Specifies the name of the HTTP tunnel transport channel.

The name field cannot contain the following characters: `# \ / , : ; " * ? < > | = + & % '`

This name must be unique across all channels in a WebSphere Application Server environment. For example, an HTTP tunnel transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type string

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type Positive integer
Default 0

HTTP transport channel settings

Use this page to view and configure an HTTP transport channel. This type of transport channel handles HTTP requests from a remote client.

An HTTP transport channel parses HTTP requests and then finds an appropriate application channel to handle the request and send a response.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** . Click on **View associated transports** for the port associated with the HTTP transport channel whose settings you want to look at.

Transport channel name

Specifies the name of the HTTP transport channel.

The name field cannot contain the following characters: `# \ / , : ; " * ? < > | = + & % '`

This name must be unique across all channels in a WebSphere Application Server environment. For example, an HTTP transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type String

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

Maximum persistent requests

Specifies the maximum number of persistent (keep-alive) requests that are allowed on a single HTTP connection. If a value of 0 (zero) is specified, only one request is allowed per connection. If a value of -1 is specified, an unlimited number of requests is allowed per connection.

Data type	Integer
Default	100

Use persistent (keep-alive) connections

When selected, the HTTP transport channel, when sending an outgoing HTTP message, uses a persistent (keep-alive) connection instead of a connection that closes after one request or response exchange occurs.

Note: If a value other than 0 is specified for the maximum persistent requests property, the Use persistent (keep-alive) connections property setting is ignored.

The default for this property is selected.

Read timeout

Specifies the amount of time, in seconds, the HTTP transport channel waits for a read request to complete on a socket after the first read request occurs. The read being waited for could be an HTTP body (such as a POST) or part of the headers if they were not all read as part of the first read request on the socket.

Data type	Integer
Default	60 seconds

Write timeout

Specifies the amount of time, in seconds, that the HTTP transport channel waits on a socket for each portion of response data to be transmitted. This timeout usually only occurs in situations where the writes are lagging behind new requests. This can occur when a client has a low data rate or the server's network interface card (NIC) is saturated with I/O.

Data type	Integer
Default	60 seconds

Persistent timeout

Specifies the amount of time, in seconds, that the HTTP transport channel allows a socket to remain idle between requests.

Data type	Integer
Default	30 seconds

Enable access and error logging
When selected, the HTTP transport channel performs NCSA access and error logging. Enabling NCSA access and error logging slows server performance.

To configure NCSA access and error logging, click **HTTP error and NCSA access logging** under **Related Items**. Even if HTTP error and NCSA access logging is configured, it is not enabled unless the Enable access and error logging property is selected.

The default value for the Enable access and error logging property is not selected.

TCP transport channel settings
Use this page to view and configure a TCP transport channels. This type of transport channel handles inbound TCP/IP requests from a remote client.

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Ports >** . Click on **View associated transports** for the port associated with the TCP transport channel whose settings you want to view.

Transport channel name
Specifies the name of the TCP transport channel.

The name field cannot contain the following characters: **# \ / , : ; ″ * ? < > | = + & % ’**

This name must be unique across all channels in a WebSphere Application Server environment. For example, an HTTP proxy inbound channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type string

Port
Specifies the TCP/IP port this transport channel uses to establish connections between a client and an application server. The TCP transport channel binds to the hostnames and ports listed for the Port property. You can specify the wildcard * (an asterisk), for the hostname if you want this channel to listen to all hosts that are available on this system. However, before specifying the wildcard value, make sure this TCP transport channel does not have to bind to a specific hostname.

Data type string

Thread pool
This field only applies for i5/OS and distributed platforms. Select from the drop-down list of available thread pools the thread pool you want the TCP transport channel to use when dispatching work.

Maximum open connections
Specifies the maximum number of connections that can be open at one time.

Data type Integer between 1 and 20,000 inclusive
Default 20,000

Inactivity timeout
Specifies the amount of time, in seconds, that the TCP transport channel waits for a read or write request to complete on a socket.

Note: The value specified for this property might be overridden by the wait times established for channels above this channel. For example, the wait time established for an HTTP transport channel overrides

the value specified for this property for every operation except the initial read on a new socket.

Data type	Integer
Default	60 seconds

Address exclude list

Lists the IP addresses that are not allowed to make inbound connections.

Use a comma to separate the IPv4 or IPv6 or both addresses to which you want to deny access on inbound TCP connection requests.

All four numeric values in an IPv4 address must be represented by a number or the wildcard character * (an asterisk).

Following are examples of valid IPv4 addresses that can be included in an Address exclude list:

```
*.1.255.0
254.*.*.9
1.*.*.*
```

All eight numeric values of an IPv6 address must be represented by a number or the wildcard character * (an asterisk). No shortened version of the IPv6 address should be used. Even though a shortened version is processed with no error given, it does not function correctly in this list. Each numeric entry should be a 1- 4 digit hexadecimal number.

Following are examples of valid IPv6 addresses that can be included in an Address exclude list:

```
0:*:*:0:007F:0:0001:0001
F:FF:FFF:FFFF:1:01:001:0001
1234:*:4321:*:9F9f:*:*:0000
```

Note: The **Address include list** and **Host name include list** are processed before the **Address exclude list** and the **Host name exclude list**. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it will not be allowed access.

Address include list

Lists the IP addresses that are allowed to make inbound connections. Use a comma to separate the IPv4 or IPv6 or both addresses to which you want to grant access on inbound TCP connection requests.

All four numeric values in an IPv4 address must be represented by a number or the wildcard character * (an asterisk).

Following are examples of valid IP addresses that can be included in an Address include list:

```
*.1.255.0
254.*.*.9
1.*.*.*
```

All eight numeric values of an IPv6 address must be represented by a number or the wildcard character * (an asterisk). No shortened version of the IPv6 address should be used. Even though a shortened version is processed with no error given, it does not function correctly in this list. Each numeric entry should be a 1- 4 digit hexadecimal number.

Following are examples of valid IPv6 addresses that can be included in an **Address include list**:

```
0:*:*:0:007F:0:0001:0001
F:FF:FFF:FFFF:1:01:001:0001
1234:*:4321:*:9F9f:*:*:0000
```

Note: The Address include list and the Host name include list are processed before the Address exclude list and the Host name exclude list. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it will not be allowed access.

Host name exclude list

List the host names that are not allowed to make connections. Use a comma to separate the URL addresses to which you want to deny access on inbound TCP connection requests.

A URL address can start with the wildcard character * (an asterisk) followed by a period; for example, `*.Rest.Of.Address`. If a period does not follow the wildcard character, the asterisk will be treated as a normal non-wildcard character. The wildcard character cannot appear any where else in the address. For example, `ibm.*.com` is not a valid hostname.

Following are examples of valid URL addresses that can be included in a Host name exclude list:

```
*.ibm.com
www.ibm.com
*.com
```

Note: The Address include list and Host name include list are processed before the Address exclude list and the Host name exclude list. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it is not allowed access.

Host name include list

Lists the host names that are allowed to make inbound connections. Use a comma to separate the URL addresses to which you want to grant access on inbound TCP connection requests.

A URL address can start with the wildcard character * (an asterisk) followed by a period; for example, `*.Rest.Of.Address`. If a period does not follow the wildcard character, the asterisk will be treated as a normal non-wildcard character. The wildcard character cannot appear any where else in the address. For example, `ibm.*.com` is not a valid hostname.

Following are examples of valid URL addresses that can be included in a hostname include list:

```
*.ibm.com
www.ibm.com
*.com
```

Note: The Address include list and Host name include list are processed before the Address exclude list and the Host name exclude list. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it is not allowed access.

DCS transport channel settings

Use this page to view and configure an DCS transport channels. This type of transport channel handles inbound Distribution and Consistency Services (DCS) messages.

By default, two channel transport chains are defined for an application server that contains a DCS channel:
- The chain named DCS contains a TCP and a DCS channel.
- The chain named DCS-Secure contains a TCP, an SSL, and a DCS channel.

Both of these chains terminate in, or use the same TCP channel instance. This TCP channel is associated with the DCS_UNICAST_ADDRESS port and is not used in any other transport chain. One instance of an SSL channel is reserved for use in the DCS-Secure chain. It also is not used in any other transport chains.

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Ports >** . Click **View associated transports** for the port associated with the DCS transport channel whose settings you want to look at.

Transport channel name
Specifies the name of the DCS transport channel.

The name field cannot contain the following characters: `# \ / , : ; " * ? < > | = + & % '`

This name must be unique across all channels in a WebSphere Application Server environment. For example, a DCS transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type String

Discrimination weight
Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type Positive integer
Default 0

SSL inbound channel
Use this page to determine which SSL inbound channel options to specify for the application server.

To view this administrative console page:
1. Click **Servers** > **Application Servers** > *server_name*.
2. Under Container settings, click **Web container transport chains** > *secure_transport_chain*.
3. Under Transport channels, click **SSL Inbound Channel (SSL_1)**.

Transport Channel Name
Specifies the name of the SSL inbound channel.

The name field cannot contain the following characters: `# \ / , : ; " * ? < > | = + & % '`

This name must be unique across all channels in an application server environment. For example, an SSL inbound channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type String

Discrimination weight
Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

Centrally managed
Specifies that the selection of an SSL configuration is based upon the outbound topology view for the Java Naming and Directory Interface (JNDI) platform.

Centrally managed configurations support one location to maintain SSL configurations rather than spreading them across the configuration documents.

Default:	Enabled

Specific to this endpoint
Specifies the SSL configuration alias that you want to use for outbound SSL communications.

This option overrides the centrally managed configuration for the JNDI (LDAP) protocol.

Session Initiation Protocol (SIP) inbound channel settings
Use this page to configure the SIP inbound channel settings.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** . Click on **View associated transports** for the port associated with the UDP transport channel whose settings you want to view.

Transport channel name
Specifies the name of the SIP inbound transport channel.

The name field cannot contain the following characters: **# \ / , : ; ″ * ? < > | = + & % ’**

This name must be unique across all channels in a WebSphere Application Server environment. For example, a SIP transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Default	UDP_(n) where (n) represents the number of instances of this channel in the system

Discrimination weight
Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	10

Session Initiation Protocol (SIP) container inbound channel settings
Use this page to configure the SIP container inbound channel settings.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** . Click on **View associated transports** for the port associated with the UDP transport channel whose settings you want to view.

Transport channel name
Specifies the name of the SIP container inbound transport channel.

The name field cannot contain the following characters: `# \ / , : ; " * ? < > | = + & % '`

This name must be unique across all channels in a WebSphere Application Server environment. For example, a SIP container transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Default UDP_(n) where (n) represents the number of instances of this channel in the system

Descrimination weight
Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type Positive integer
Default 10

User Datagram Protocol (UDP) Inbound channel settings
Use this page to configure the UDP Inbound channel settings.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** . Click on **View associated transports** for the port associated with the UDP transport channel whose settings you want to view.

Transport channel name
Specifies the name of the UDP inbound transport channel.

The name field cannot contain the following characters: `# \ / , : ; " * ? < > | = + & % '`

This name must be unique across all channels in a WebSphere Application Server environment. For example, a UDP transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Default UDP_(n) where (n) represents the number of instances of this channel in the system

Address exclude list
Specifies the IP addresses that are not allowed to make inbound connections. Use a comma to separate the IPv4 and/or IPv6 addresses to which you want to deny access on inbound UDP connection requests.

The address include list and host name include list are processed before the address exclude list and the host name exclude list. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it is not allowed access.

Data type String

Range Valid IPv4 and IPv6 addresses with a wildcard character (*), an asterisk. All four elements of an IPv4 address must be represented by a number or a wildcard character. All eight numeric values of an IPv6 address must be represented by a number or the wildcard character (*).

Example The following examples are valid IPv4 addresses that can be included in an Address exclude list:

```
*.1.255.0
254.*.*.9
1.*.*.*
```

All eight numeric values of an IPv6 address must be represented by a number or the wildcard character (*), an asterisk. No shortened version of the IPv6 address should be used. Even though a shortened version is processed with no error given, it does not function correctly in this list. Each numeric entry should be a 1- 4 digit hexadecimal number. The following examples are valid IPv6 addresses that can be included in an Address exclude list:

```
0:*:*:0:007F:0:0001:0001
F:FF:FFF:FFFF:1:01:001:0001
1234:*:4321:*:9F9f:*:*:0000
```

Address include list

Specifies the IP addresses that are allowed to make inbound connections. Use a comma to separate the IPv4 and/or IPv6 addresses to which you want to allow access on inbound UDP connection requests.

Data type String

Range Valid IPv4 and IPv6 addresses with a wildcard character (*), an asterisk. All four elements of an IPv4 address must be represented by a number or a wildcard character (*). All eight numeric values of an IPv6 address must be represented by a number or the wildcard character (*).

Web container inbound transport channel settings

Use this page to view and configure a Web container inbound channel transport. This type of channel transport handles inbound Web container requests from a remote client.

To view this administrative console page, click **Servers** > **Application servers** > *server_instance* > **Container Settings** > **Web Container Settings**> **Web container transport chains** > *transport_chain* > **Web container inbound channel (***transport_channel_name***)** .

Transport Channel Name

Specifies the name of the Web container inbound transport channel.

The name field cannot contain the following characters: **# \ / , : ; ″ * ? < > | = + & % ’**

This name must be unique across all channels in a WebSphere Application Server environment. For example, a Web container inbound transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type String

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

Write buffer size

Specifies the amount of content in bytes to buffer unless the servlet explicitly calls flush/close on the response/writer output stream.

Data type	bytes
Default	9192 bytes

Developing custom services

A custom service provides the ability to plug into a WebSphere Application Server application server to define a hook point that runs when the server starts and shuts down. An application server configuration provides settings that control how an application server provides services for running applications and their components.

A developer implements a custom service containing a class that implements a particular interface. The administrator configures the custom service in the administrative console, identifying the class created by the developer. When an application server starts, any custom services defined for the application server are loaded and the server runtime calls their initialize methods.

To define a hook point to be run when a server or node agent starts and shuts down, you develop a custom service class and then configure a custom service instance. When the application server or node agent starts, the custom service starts and initializes.

The following restrictions apply to the WebSphere Application Server custom services implementation:

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

These restrictions apply to the shutdown and init methods equally. Some JNDI operations are available.

1. Develop a custom service class that implements the com.ibm.websphere.runtime.CustomService interface. The properties passed by the application server runtime to the initialize method can include one for an external file containing configuration information for the service. You can use the

externalConfigURLKey property to retrieve this information. In addition, these properties can contain any name-value pairs that are stored for the service, along with the other system administration configuration data for the service. The properties are passed to the initialize method of the service as a Properties object.

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

2. Configure the custom service.

   In the administrative console, click **Servers > Application Servers**, and then under Server Infrastructure, click **Custom Services > New**. Then, on the settings page for a custom service instance, create a custom service configuration for an existing application server or node agent, supplying the name of the class implemented. If your custom service class requires a configuration file, specify the fully-qualified path name to that configuration file in the **externalConfigURL** field. This file name is passed into your custom service class.

   To invoke a native library from the custom service, provide the path name in the **Classpath** field in addition to the path names that are used to locate the classes and JAR files for the custom service. Doing this adds the path name to the WebSphere Application Server extension classloader, which allows the custom service to locate and correctly load the native library.

3. Stop the application server and restart it.

   If you are developing a custom service for an application server, stop the application server and then restart the server.

   If you are developing a custom service for a node agent, stop and then restart the processing of the node agent. In the administrative console, click **System Administration > Node Agents**, and place a checkmark in the check box beside the node agent you want to stop, then click **Stop**. To restart the node agent, place a checkmark in the check box beside the node agent, then click **Restart**.

4. Ensure the initialize and shutdown methods of the custom service perform as intended.

   Check the application server or node agent to ensure that the initialize method of the custom service ran as intended. Also ensure that the shutdown method performs as intended when the server or node agent stops.

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

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

    static String ConfigFileName="";

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

```
        }

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

Custom service collection

Use this page to view a list of services available to the application server and to see whether the services are enabled. A custom service provides the ability to plug into a WebSphere application server and define code that runs when the server starts or shuts down.

External Configuration URL

Specifies the URL for a custom service configuration file.

If your custom services class requires a configuration file, the value provides a fully-qualified path name to that configuration file. This file name is passed into your custom service class.

Classname

Specifies the class name of the service implementation. This class must implement the Custom Service interface.

Display Name

Specifies the name of the service.

Enable service at server startup

Specifies whether the server attempts to start and initialize the service when its containing process (the server) starts. By default, the service is not enabled when its containing process starts.

Custom service settings

Use this page to configure a service that runs in an application server.

Enable service at server startup:

Specifies whether the server attempts to start and initialize the service when its containing process (the server) starts. By default, the service is not enabled when its containing process starts.

Data type	Boolean
Default	false

External Configuration URL:

Specifies the URL for a custom service configuration file.

If your custom services class requires a configuration file, specify the fully-qualified path name to that configuration file for the value. This file name is passed into your custom service class.

Data type	String
Units	URL

Classname:

Specifies the class name of the service implementation. This class must implement the Custom Service interface.

Data type	String
Units	Java class name

Display Name:

Specifies the name of the service.

Data type	String

Description:

Describes the custom service.

Data type	String

Classpath:

Specifies the class path used to locate the classes and JAR files for this service.

Data type	String
Units	Class path

Defining application server processes

To enhance the operation of an application server, you can define command-line information for starting or initializing an application server process. Such settings define runtime properties such as the program to run, arguments to run the program, and the working directory.

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

1. In the administrative console, click **Servers > Application Servers** click on an application server name, and then click **Java and Process Management > Process Definition**.

   You can also define application server processes using the wsadmin tool. For more information, see the *Using the administrative clients* PDF.
2. On the settings page for a process definition, specify the name of the executable to run, any arguments to pass when the process starts running, and the working directory in which the process will run. Then click **OK**.
3. Specify process execution statementsfor starting or initializing a UNIX or i5/OS process.
4. Specify monitoring policies to track the performance of a process.
5. Specify process logs to which standard out and standard error streams write. Complete this step if you do not want to use the default file names.
6. Specify name-value pairs for properties needed by the process definition.

   Important: Each custom property name must be unique. If the same name is used for multiple properties, the process uses the value specified for the first property that has that name.
7. Stop the application server and then restart the server.
8. Check the application server to ensure that the process definition runs and operates as intended.

Process definition settings

Use this page to configure a process definition. A process definition includes the command line information necessary to start or initialize a process.

To view this administrative console page, click **Servers > Application Servers >** *server_name*. Then under Server Infrastructure click **Java™ Process Management > Process Definition**.

Executable Name

This command line information specifies the executable name that is invoked to start the process.

Data type String

Executable Arguments

This command line information specifies the arguments that are passed *arg1 arg2 arg3*.

Data type String
Units Java command-line arguments

Start Command (startCommand)

This command line information specifies the platform-specific command to launch the server process.

Start Command Args (startCommandArgs)

This command line information specifies any additional arguments required by the start command.

Stop Command (stopCommand)

This command line information specifies the platform-specific command to stop the server process

Specify two commands in the field, one for the Stop command and one for the Immediate Stop (CANCEL) command.

Data type String
Format STOP *server_short_name*;CANCEL *server_short_name*

Stop Command Args (stopCommandArgs)

This command line information specifies any additional arguments required by the stop command.

Specify arguments for the Stop command and the Immediate Stop (CANCEL) command.

Data type String
Format *stop command arg string*;*immediate stop command arg string*
Example ;ARMRESTART

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

Terminate Command (terminateCommand)

This command line information specifies the platform-specific command to terminate the server process.

Data type String
Format FORCE *server_short_name*

Terminate Command Args (terminateCommandArgs)
This command line information specifies any additional arguments required by the terminate command.

The default is an empty string.

Data type	String
Format	*terminate command arg string*

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

Data type	String

Executable target type
Select whether the executable target is a Java class or an executable JAR file.

Executable target
Specifies the name of the executable target. If the target type is a Java class name, this field contains the main() method. If the target type is an executable JAR file, this field contains the name of that JAR file.

Data type	String

Process execution settings
Use this page to view or change the process execution settings for a server process that applies to either an application server, a node agent or a deployment manager.

To view this administrative console page for an application server, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Java and Process Management > Process Execution**.

To view this administrative console page for a node agent, click **System Administration > Node agents >***node_agent_name*. Then, under Server Infrastructure , click **Java and Process Management > Process Definition > Process Execution**.

To view this administrative console page for a deployment manager, click **System Administration > Deployment manager**. Then, under Server Infrastructure, click **Java and Process Management > Process Definition > Process Execution**.

Process Priority:

Specifies the operating system priority for the process. The administrative process that launches the server must have root operating system authority in order to honor this setting.

Data type	Integer
Default	20 for WebSphere Application Server on all operating systems.

UMASK:

Specifies the user mask under which the process runs (the file-mode permission mask).

Data type	Integer

Run As User:

Specifies the user that the process runs as.

Data type String

For i5/OS, additional steps are required in order to run as a userid other than QEJBSVR. For more information, see the Security section of the WebSphere Application Server for iSeries online documentation. Go to http://www-1.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/index.html and navigate to the WebSphere Application Server for iSeries Security information.

Run As Group:

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

On i5/OS, the Run As Group setting is ignored.

Data type String

Process logs settings

Use this page to view or change settings for specifying the files to which standard out and standard error streams write.

To view this administrative console page, in the administrative console:

For an application server on an i5/OS or distributed platform, click **Servers** > **Application Servers** > *server_name*, and then, under Server Infrastructure, click **Java and Process Management** > **Process Definition** > **Process Logs**.
For a deployment manager on an i5/OS or distributed platform, click **System Administration** > **Deployment Manager**, and then under Server Infrastructure, click **Java and Process Management** > **Process Definition** > **Process Logs**.

Stdout File Name:

Specifies the file to which the standard output stream is directed. The file name can include a symbolic path name defined in the variable entries.

Use the field on the configuration tab to specify the file name. Use the field on the Runtime tab to select a file for viewing. View the file by clicking **View**.

Direct server output to the administrative console or to the process that launched the server, by either deleting the file name or specifying `console` on the configuration tab.

Data type String
Units File path name

Stderr File Name:

Specifies the file to which the standard error stream is directed. The file name can include a symbolic path name defined in the variable entries.

Use the field on the configuration tab to specify the file name. Use the field on the runtime tab to select a file for viewing. View the file by clicking **View**.

Data type String

Units File path name

Monitoring policy settings

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

To view this administrative console page, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Java and Process Management > Monitoring Policy**.

Maximum Startup Attempts:

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

Data type Integer

Ping Interval:

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

Data type Integer
Range Set the value greater than or equal to 0 (zero) and less than 2147483647. If you specify a value greater than 2147483647, the application server acts as though you set the value to 0.

Ping Timeout:

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

Data type Integer
Units Seconds
Range Set the value greater than or equal to 0 (zero) and less than 2147483647. If you specify a value greater than 2147483647, the application server acts as though you set the value to 0.

Automatic Restart:

Specifies whether the process should restart automatically if it fails. On distributed systems, the default is to restart the process automatically. On a z/OS system, the default is to not start the process automatically.

If you change the value specified for this field, you must restart the application server and the node agent before the new setting takes effect.

This setting does not affect what you specified for the Node Restart State setting. The two settings are mutually exclusive.

Data type Boolean

Default	true (distributed) / false (z/OS)

Node Restart State:

Specifies the desired behavior of the servers after the node completely shuts down and restarts.

If a server is already running when the node agent stops, that server is still running after the node agent restarts. If a server is stopped when the node agent restarts, whether the node agent starts the server depends on the setting for this property:

- If this property is set to STOPPED, node agent does not start the server.
- If this property is set to RUNNING, the node agent always starts the server.
- If this property is set to PREVIOUS, the node agent starts the server only if the server was running when the node agent stopped.

This setting does not affect what you specified for the Automatic Restart setting. The two settings are mutually exclusive.

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

Automatically restarting server processes

There are several server processes related to WebSphere Application Server products that the operating system can monitor and automatically restart when the server processes stop abnormally. This task describes how to set up these *monitored* processes.

To set up this function on a Linux or UNIX-based operating system, you must have root authority to edit the inittab file.

To set up this function on a Microsoft Windows operating system, you must belong to the Administrator group and have the following advanced user rights:

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

The Installation wizard grants you the user rights if your user ID is part of the administrator group.

If you are running on a Microsoft Windows Operating System, the Installation wizard displays a message that states that although the advanced user rights are now effective, they do not display as effective until the next time you log on to the Windows machine.

You can also add the advanced user rights manually if you are performing a silent installation on a Windows operating system. For example, to grant the user rights to your administrator group user ID on a Windows operating system, perform the following procedure:

1. Click **Administrative Tools** in the Control Panel.
2. Click **Local Security Policy**.
3. Click **Local Policies**.
4. Click **User Rights Assignments**.
5. Right click **Act as part of the operating system**.
6. Click **Security**.

7. Click **Add**.
8. Click your user ID.
9. Click **Add**.
10. Click **OK**.
11. Click **OK**.
12. Right click **Log on as a service**.
13. Click **Security**.
14. Click **Add**.
15. Click **OK**.
16. Click **OK**.
17. Reboot your machine to make the settings effective.

Consult your Windows help system for more information.

There are several environments where you might use this function of automatically restarting servers. You can restart the ***server1*** managed node process, for example. Here is a list of processes you might consider restarting:
- The **server1** managed node process
- The **server1** process on a stand-alone Application Server
- The **dmgr** process on a deployment manager node
- The **nodeagent** server process on any managed node
- The **IBM HTTP Server** process
- The **IBM HTTP Administration** process

On a Windows operating system, you can create Windows services during installation, using the installation wizard. Each Windows service controls a single process, such as a stand-alone WebSphere Application Server instance. Multiple stand-alone Application Server processes require multiple Windows services, which you can define. The wizard lets you create services for these servers:
- The **server1** managed node process, defined as a manually started (versus automatic) service
- The **server1** stand-alone Application Server process, defined as a manually started service
- The **IBM HTTP Server** process and the **IBM HTTP Administration** process, defined as automatically started services when you choose to install the IBM HTTP Server feature
- The **dmgr** process on a deployment manager node, defined as a manually started service

The installation wizard does not provide a way to create a service for a node agent because the deployment manager instantiates each node agent after installation when you add an Application Server node to the deployment manager cell. For this reason, you must manually create a function that automatically starts a failed node agent server process.

On a Linux or UNIX-based operating system, you must manually create a shell script that automatically starts any of the processes previously mentioned. Each UNIX shell script controls a single process, such as a stand-alone WebSphere Application Server instance. Multiple stand-alone Application Server processes require multiple UNIX scripts, which you can define.

In a Network Deployment environment, the **addNode** or **startNode** command starts a single unmonitored node agent only, the nodeagent process, and does not start all of the processes that you might define on the node. While running, the node agent monitors and restarts Application Server processes on that node, on either a Windows or a Linux and UNIX-based platform. Each Application Server process has MonitoringPolicy configuration settings that the node agent uses when monitoring and restarting the process.

It is recommended that you manually set up a monitored process for the deployment manager dmgr server and for any node agent defined for your system. To set up a monitored process:

- On a Windows operating system, use a Windows service. You can install the WebSphere Application Server Network Deployment product as a Windows service during installation, or at a later time
- On a Linux or UNIX-based operating system, use the rc.was example shell script that is provided with the Linux and UNIX versions of the product.

1. On a Windows operating system, **Use the Profile Management tool** to set up a Windows service to automatically monitor and restart processes related to the WebSphere Application Server product.
   - Perform the following procedure from the Profile Management tool to select services that the installation wizard can set up:
     a. Click **Run WebSphere Application Server Network Deployment as a service**.

        If you select this option, the installation wizard creates the following service during installation:

        IBMWAS6Service - *node_name*

        IBMWAS6Service - *node_name* service controls the *node_name* process.

        After you complete and verify the installation, use the Windows Services panel to change the **IBMWAS6Service -** *node_name* service to an automatic startup type.
        1) Right click **IBMWAS6Service -** *node_name* and click **Properties**.
        2) Click **Automatic** from the **Startup type** list box and click **OK**.

     b. Click **Run IBM HTTP Server as a service**.

        Select this option on the machine where you are installing the IBM HTTP Server.

        If you select this option, the installation wizard creates the following services during the installation:
        – **IBM HTTP Server 2.0.*x***
        – **IBM HTTP Administration 2.0.*x***

        The installation wizard defines the startup type of these services as **automatic**. It is not necessary for you to change the type from manual to automatic.

     c. Enter your user ID and password and click **Next**.

   In a coexistence environment, you can change the default service names to make them unique. In a same version coexistence scenario for IBM HTTP Server 2.0.x on a Windows platform, you cannot use the default service names created by the installer because they are common.

   To work around this problem:
   a. Install the first copy of IBM HTTP Server, either by itself or with WebSphere Application Server and select to install the services.
   b. Customize the service names for the first install by running the following commands from the first install location:

      ```
      apache -k install -n "IHS 2.0(1)"
      apache -k install -f conf\admin.conf -n "IHS 2.0 Administration (1)"
      ```

   c. Edit the AdminAlias directive in the *installLocation 1*\conf\admin.conf file to point to the new service name, such as **IHS 2.0(1)**.
   d. Remove the default service names installed by the first install by running the following commands:

      ```
      apache -k uninstall -n "IBM HTTP Server 2.0"
      apache -k uninstall -n "IBM HTTP Administration 2.0"
      ```

   e. Install the second copy of IBM HTTP Server, either by itself or with WebSphere Application Server. The default service names correspond to the second install.

   Note: Customized service names must be unique on your system.

2. On a Linux or UNIX operating system, after you install the WebSphere Application Server product, set up a Linux and UNIX-based shell script to automatically monitor and restart the node agent process or any other related server process.
   a. Locate the rc.was example shell script, which is in the *app_server_root*/bin directory.
   b. Create a new shell script for each process that the operating system is to monitor and restart.
   c. Edit each shell script according to comments in its header, which provide instructions for identifying a WebSphere Application Server process.

d. Edit the inittab table of the operating system, to add an entry for each shell script you have created.

   Comments in the header of the rc.was file show a sample inittab entry line for adding the script. This inittab entry causes the Linux and UNIX-based system to call each shell script whenever the system initializes. As it runs, each shell script monitors and starts the server process you specified.

Each shell script monitors and restarts the following processes in an WebSphere Application Server Network Deployment environment:

- A server process on a managed node
- A node agent process on a managed node
- A stand-alone Application Server process
- A deployment manager process

On a Windows operating system, you can

- Use the **net start** and **net stop** commands to control the IBM HTTP Server services on a Windows system. For more information about these commands, see the Windows help file. Access these commands from the Start menu, clicking **Start > Programs > IBM HTTP Server**.
- Use the **Start the Server** and **Stop the Server** commands to control the IBM WebSphere Application Server process. Access these commands from the Start menu, clicking **Start > Programs > IBM WebSphere > Application Server V6**.
- Use the **Start the Manager** and **Stop the Manager** commands to control the Network Deployment dmgr process. Access these commands from the Start menu, clicking **Start > Programs > IBM WebSphere > Application Server V6 > Deployment Manager**.
-

Processes started by a **startServer** command, a **startNode** command, or a **startManager**command are not running as monitored processes, regardless of how they are configured.

For example, you can configure a server1 process as a monitored process. However, if you start the server1 process using the **startServer** command, the operating system does not monitor or restart the server1 process because the operating system did not originally start the process as a monitored process.

After the process is set up, the operating system can monitor each server process and restart the process if it stops.

Return to Defining application server processes to continue.

Configuring the JVM

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

The Java virtual machine (JVM) is an interpretive computing engine responsible for running the byte codes in a compiled Java program. The JVM translates the Java byte codes into the native instructions of the host machine. The application server, being a Java process, requires a JVM in order to run, and to support the Java applications running on it. JVM settings are part of an application server configuration.

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

1. In the administrative console, click **Servers > Application Servers >***server***> Java and Process Management > Process Definition > Java Virtual Machine**.
2. Specify values for the JVM settings as needed and click **OK**.
3. Click **Save** on the console task bar.

4. Restart the application server.

"Configuring application servers for UCS Transformation Format" on page 209 provides an example that involves specifying a value for the **Generic JVM Arguments** property on the Java virtual machine page to enable UTF-8 encoding on an application server. Enabling UTF-8 allows multiple language encoding support to be used in the administrative console.

"Configuring JVM sendRedirect calls to use context root" on page 289 provides an example that involves defining a property for the JVM.

Caching classes previously loaded by a user class loader

Caching classes previously loaded by the i5/OS Java virtual machine (JVM) user class loaders can reduce the amount of time that it takes to load servlets, JavaServer Page (JSP) files and enterprise beans. Caching classes is not required for most applications. However, in some situations, caching classes improves an application's performance.

Application server class loaders, application class loaders, and application module class loaders load WebSphere Application Server components such as servlets, JSP files and enterprise beans. These class loaders are called *user class loaders* in the i5/OS documentation.

If servlets, JSP files or enterprise beans in your application are slow to load, a cache is available to improve the startup performance of these components. The cache enables the i5/OS JVM to identify classes previously loaded with user class loaders.

When the cache is enabled, each Java program object (JVAPGM) created by a user class loader is cached for reuse during the initial class loading. The first time the class is loaded, the loading is slow because JVAPGMs are created and bytecode is verified during the loading. After a class is in the cache, JVAPGM creation and bytecode verification does not occur, which makes subsequent loadings much quicker. You must run your application to initially add the classes to the cache.

Before attempting to enable the cache, determine whether using the cache might enhance the performance of your components. You should only enable the cache if your components are slow to load in the following situations:

Your application contains these components	The components load at this time	Comments
Servlets that are configured to load when an application server starts or enterprise beans	When the application server starts up	Having a large number of these components, or complex components that access many classes in your application, slows application server startup.
Servlets that are not configured to load when an application server starts or JSP files	When classes of the component are first accessed	If these components are complex or access many classes in your application, it takes longer to load these components for the first time.

To enable the user class loader cache:

1. In the administrative console, click **Servers > Application Servers >** *server_name* **> Process Definition > Java Virtual Machine > Custom Properties**.
2. Under Server Infrastructure, click **Java and Process Management > Process Definition >** *process_name* **> Java Virtual Machine**.
3. Under Additional Properties, click **Custom Properties > New**.
4. Specify os400.define.class.cache.file in the Name field and the full path name of a valid Java archive (JAR) file to hold the Java Program objects (JVAPGMs) in the Value field. The os400.define.class.cache.file custom property enables the Java user class loader cache. The JAR file specified in the Value field must contain a valid JAR entry, but does not have to have any other content beyond the single member required to make the JAR command function.

The /QIBM/ProdData/Java400/QDefineClassCache.jar cache JAR file is shipped with WebSphere Application Server. You can use this JAR file as is, or copy and rename it. You can also create your own cache JAR file:

a. Enter the STRQSH command to start the Qshell.
b. Switch to the directory where you want the JAR file located. Make the directory first, if necessary.
   ```
   mkdir /cache cd /cache
   ```
c. Create a dummy file to place in the JAR. Use any name. This example uses example:
   ```
   touch example
   ```
d. Build the JAR file. This example names the JAR file MyAppCache.jar:
   ```
   jar -cf MyAppCache.jar example
   ```
e. Clean up the dummy file:
   ```
   rm example
   ```

This JAR file must not be on any classpath.

You can use DSPJVAPGM on this JAR file to determine how many JVAPGMs are cached. The **Java programs** field of the DSPJVAPGM display indicates how many JVAPGMs are cached, and the **Java program size** field indicates how much storage is consumed by cached JVAPGMs. Other fields of the display are meaningless when DSPJVAPGM is applied to a JAR file used for caching.

You can also use CHGJVAPGM on the JAR file to change the optimization of the classes in the cache. CHGJVAPGM only affects programs currently in the cache. Classes added to the cache are optimized according to the other properties described in this list.

Name	Example value
os400.define.class.cache.file	/QIBM/ProdData/Java400/QDefineClassCache.jar

5. Click **OK**.
6. **Optional:** Click **New** again and specify one of the following custom properties to customize the user class loader cache. Repeat this step as many times as necessary to complete your customization.

   os400.define.class.cache.hours
   Optionally, specify the number of hours an unused JVAPGM persists in the cache. When a JVAPGM has not been used and this timeout is reached, the JVAPGM is removed from the cache. The default value is 168 (one week). The maximum value is 9999 (about 59 weeks).

   os400.define.class.cache.maxpgms
   Optionally, specify the maximum number of JVAPGMs the cache can hold. If this value is reached, the least recently used JVAPGMs are replaced first. The default value is 5000. The maximum value is 40000.

   For example, to use the shipped cache JAR with a maximum of 10,000 JVAPGMs that have a maximum life of 1 year, add the specify the following custom propertie:

Name	Value
os400.define.class.cache.hours	8760
os400.define.class.cache.maxpgms	10000

   Other Java system properties, such as os400.defineClass.optLevel, can be used to customize how JVAPGMs are created in the cache.
7. Click **OK** and then click **Save** to save the configuration changes.
8. Restart the application server.

Classes will be cached after they are initially loaded by the i5/OS Java virtual machine (JVM) user class loaders.

Running classes using JVM direct execution

You can configure an application server to run classes directly instead of using the just-in-time (JIT) mode.

A class loader loads WebSphere Application Server components such as servlets, JavaServer Pages, and enterprise beans. These components do not run classes directly. The direct execution (DE) capabilities of the i5/OS Java virtual machine (JVM) can be used to run classes directly.

When WebSphere Application Server component code is running, Java program (JVAPGM) objects are not used. Classes that a JVM class loader or the WebSphere Application Server extensions class loader load use DE capabilities and JVAPGM objects. Java caching enables these components to use permanent JVAPGM objects and DE capabilities, even though most applications do not require this functionality.

By default, application servers run with the java.compiler Java system property set to jitc_de. This value indicates that JIT compilation is done for any Java object for which a JVAPGM object does not exist (or cannot be used, as is the case for WebSphere Application Server application class loaders). This setting provides the best overall performance.

You can configure application servers to use direct execution instead of JIT compilation. Doing so results in longer startup times, because creating a JVAPGM takes longer than creating JIT stubs. Because of performance improvements in the JIT run time, performance of direct execution even after JVAPGM creation is probably slower.

To configure an application server to use direct execution for all classes:

1. In the administrative console, click **Servers > Application Servers >** *server_name*.
2. Under Server Infrastructure, click **Java and Process Management > Process Definition >** *process_name* **> Java Virtual Machine**.
3. Under Additional Properties, click **Custom Properties > New**.
4. Specify one of the following for the Name and Value of the new custom property:
   - os400.defineClass.optLevel in the Name field and a direct execution optimization level in the Value field. (Specify 0 for interpret, or specify direct execution optimization levels of 10, 20, 30, or 40.
   - java.compiler in the Name field and NONE in the Value field. (This combination makes the application server use full interpretation for all WebSphere Application Server components.)
5. Click **OK** and then click **Save** to save the configuration changes.
6. Restart the application server.

You can now use JVM direct execution to run classes.

Java virtual machine settings

Use this page to view and change the Java virtual machine (JVM) configuration settings of a process for an application server.

To view this administrative console page, connect to the administrative console and navigate to the Java virtual machine panel:

For the i5/OS and distributed platforms ND configuration:

Application server	**Servers > Application Servers >** *server1* **> Java and Process Management > Process Definition > Java Virtual Machine**
Deployment manager	**System Administration > Deployment Manager > Java and Process Management > Process definition > Java Virtual Machine**

Node agent	**System Administration > Node Agent >** *nodeagent* **> Java and Process Management > Process definition > Java Virtual Machine**

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

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

Data type String
Units Class path

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

Data type String

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

Data type Boolean
Default false

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

Data type Boolean
Default false

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

Data type Boolean
Default false

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

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

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

Important: For i5/OS, the minimum heap size must always be less than the maximum heap size. Never set the minimum heap size and maximum heap size properties to the same value.

Data type Integer
Default For i5/OS, the default is 96

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

Increasing the heap size can improve startup. By increasing heap size, you can reduce the number of garbage collection occurrences with a 10% gain in performance.

Increasing the size of the Java heap usually improves throughput until the heap no longer resides in physical memory. When the heap size exceeds the physical memory, the heap begins swapping to disk which causes Java performance to drastically decrease. Therefore, it is important to set the maximum heap size to a value that allows the heap to be contained within physical memory.

To prevent paging, you should allow a minimum of 256MB of physical memory for each processor and 512 MB of physical memory for each application server. If possible, adjust the available memory when paging occurs if processor utilization is low because of this paging.

For i5/OS, when this property is set to 0 (zero), the garbage collector runs only when the garbage collector threshold has been reached. When a value other than 0 is specified, the garbage collector runs whenever the heap size reaches the specified maximum size. However, unlike a normal garbage collector, if the maximum size is reached, all application threads must wait until the garbage collection process has finished before they can continue running. This might cause undesirable pause times. Therefore, you should use the maximum heap size as a safety net to handle times of unexpected heap growth and to ensure that the heap doesn't grow larger than the available memory. Under normal circumstances, the specified maximum heap size should never be reached.

Data type Integer
Default For i5/OS, the default is 0, which indicates that no maximum heap size is set.

Run HProf
Specifies whether to use HProf profiler support. To use another profiler, specify the custom profiler settings using the HProf Arguments setting. The default is not to enable HProf profiler support.

If you set the Run HProf property to true, then you must specify command-line profiler arguments as values for the HProf Arguments property.

Data type Boolean
Default false

HProf arguments
Specifies command-line profiler arguments to pass to the JVM code that starts the application server process. You can specify arguments when HProf profiler support is enabled.

HProf arguments are only required if the Run HProf property is set to true.

Data type String

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

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

Data type	Boolean
Default	false

Debug arguments

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

For WebSphere Application Server Network Deployment configurations, Debug arguments are only required if the Debug Mode property is set to true. If you enable debugging on multiple application servers on the same node, make sure that the servers are using different `address` arguments, which define the port for debugging. For example, if you enable debugging on two servers and leave the default debug port for each server as `address=7777`, the servers could fail to start properly.

Data type	String
Units	Java command-line arguments

Generic JVM arguments

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

The following are optional command line arguments that you can ente int the Generic JVM arguments field. If you enter more than one argument, enter a space between each argument.

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

- **-Dcom.ibm.CORBA.RequestTimeout=*timeout_interval***

  You can use **-Dcom.ibm.CORBA.RequestTimeout=*timeout_interval*** to specify the timeout period for responding to requests sent from the client. This argument uses the -D option. *timeout_interval* is the timeout period in seconds. If your network experiences extreme latency, specify a large value to prevent timeouts. If you specify a value that is too small, an application server that participates in workload management can time out before it receives a response.

  Be careful specifying this property; it has no recommended value. Set it only if your application is experiencing problems with timeouts.

- **-Dcom.ibm.websphere.wlm.unusable.interval=*interval***

  You can use the **-Dcom.ibm.websphere.wlm.unusable.interval=*interval*** argument to change the value for the com.ibm.websphere.wlm.unusable.interval property if the workload management state of the client is refreshing too soon or too late. This property specifies the time interval that the workload management client run time waits after it marks a server as unavailable before it attempts to contact the server again. This argument uses the -D option. *interval* is the time in seconds between attempts. The default value is 300 seconds. If the property is set to a large value, the server is marked as unavailable for a long period of time. This prevents the workload management refresh protocol from refreshing the workload management state of the client until after the time period has ended.

Data type	String
Units	Java command line arguments

Executable JAR file name

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

Data type	String
Units	Path name

Disable JIT
Specifies whether to disable the just in time (JIT) compiler option of the JVM code.

If you disable the JIT compiler, throughput decreases noticeably. Therefore, for performance reasons, keep JIT enabled.

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

Operating system name
Specifies JVM settings for a given operating system.

For the Network Deployment product, when the process starts, the process uses the JVM settings for the node as the JVM settings for the operating system.

Data type	String

Configuring JVM sendRedirect calls to use context root
If the com.ibm.websphere.sendredirect.compatibility property is not set and your application servlet code has statements such as *sendRedirect("/home.html")*, your Web browser might display messages such as *Error 404: No target servlet configured for uri: /home.html*.

transition: The com.ibm.websphere.sendredirect.compatibility property is deprecated. You should modify your applications to redirect non-relative URLs (those starting with a "/") relative to the servlet container (*web_server_root*) instead of relative to the Web application context root.

To instruct the server not to use the Web server's document root and to use instead the Web application's context root for sendRedirect() calls, configure the JVM by setting the com.ibm.websphere.sendredirect.compatibility property to a `true` or `false` value.

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

Setting custom JVM properties
You can use the administrative console to change the values of JVM custom properties.

To set custom properties, connect to the administrative console and navigate to the Java virtual machine custom properties panel.

Application server	

Deployment manager	
Node agent	

If the custom property is not present in the list of already defined custom properties, create a new property, and enter the property name in the Name field and a valid value in the Value field. Restart the server to complete your changes.

com.ibm.websphere.network.useMultiHome

Set this property in a multihomed environment where WebSphere Application Server is restricted to listen only on a specific IP address for Discovery and SOAP messages. The settings for the **com.ibm.websphere.network.useMultiHome** property are as follows:

- Setting this property to `false` specifies that WebSphere Application Server will listen on all IP addresses on the host for Discovery and SOAP messages.
- Setting this property to `true` specifies that WebSphere Application Server will only listen on the configured host name for Discovery and SOAP messages. If you set this property to `true`, you should have a host name configured on WebSphere Application Server that resolves to a specific IP address.
- Setting this property to `null` specifies that WebSphere Application Server will only listen on the default IP address only.

If you cannot contact the server, check the setting for **com.ibm.websphere.network.useMultihome** to ensure it is correct. You can change the value through the administrative console. Modify the defaults by setting the value for the server, deployment manager, and node agent. In order for these changes to take place, you must restart the server.

com.ibm.websphere.deletejspclasses

Deletes JavaServer Pages classes for all applications after those applications have been deleted or updated. By default, the value of this property is `true`.

com.ibm.websphere.deletejspclasses.delete

Deletes JavaServer Pages classes for all applications after those applications have been deleted, but not after they have been updated. By default, the value of this property is `true`.

com.ibm.websphere.deletejspclasses.update

Deletes JavaServer Pages classes for all applications after those applications have been updated, but not after they have been deleted. By default, the value of this property is `true`.

java.net.preferIPv4Stack

The **java.net.preferIPv4Stack** custom property disables IPv6 support. On operating systems where IPv6 support is available, the underlying native socket that WebSphere Application Server uses is an IPv6 socket. On IPv6 network stacks that support IPv4-mapped addresses, you can use IPv6 sockets to connect to and accept connections from both IPv4 and IPv6 hosts.

Setting this property to **true** disables the dual mode support in the JVM which might, in turn, disrupt normal WebSphere Application Server functions. Therefore, it is important to understand the full implications before using this property. In general, setting this property is not recommended.

The default value for this custom property is false, except on the Windows operating system where the default is true.

com.ibm.websphere.management.registerServerIORWithLSD

The **com.ibm.websphere.management.registerServerIORWithLSD** custom property is used to control whether a federated server registers with the Location Service Daemon (LSD). Normally, a federated server requires the node agent to be running. To direct the server to not register with the LSD and remove its dependency on an active node agent, the **com.ibm.websphere.management.registerServerIORWithLSD** JVM custom property must be set to `false`, and the ORB_LISTENER_ADDRESS must be set to a value greater than 0 so that the ORB listens at a fixed port. The setting for this property is ignored if the ORB_LISTENER_ADDRESS property is set to 0 (zero) or is not specified, and the federated server registers with the LSD.

Set this property to `false` if you want the server to run even when the node agent is not running. When this property is set to `true`, the federated server registers with the LSD.

The default value for this custom property is `true`.

invocationCacheSize

The invocationCacheSize custom property is used to control the size of the invocation cache. The invocation cache holds information for mapping request URLs to servlet resources. A cache of the requested size is created for each worker thread that is available to process a request. The default size of the invocation cache is 50. If more than 50 unique URLs are actively being used (each JavaServer Page is a unique URL), you should increase the size of the invocation cache.

A larger cache uses more of the Java heap, so you might also need to increase the maximum Java heap size. For example, if each cache entry requires 2KB, maximum thread size is set to 25, and the URL invocation cache size is 100; then 5MB of Java heap are required.

You can specify any number higher than 0 for the cache size. Setting the value to zero disables the invocation cache.

Preparing to host applications

Rather than use the default application server provided with the product, you can configure a new server and set of resources.

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

1. Create an application server.
2. Create a virtual host.
3. Configure a Web container. See the *Administering applications and their environment* PDF for more information.
4. Configure an EJB container. See the *Administering applications and their environment* PDF for more information.
5. Create resources for data access. See the *Administering applications and their environment* PDF for more information.
6. Create a JDBC provider and data source. See the *Administering applications and their environment* PDF for more information.
7. Create a URL and URL provider. See the *Administering applications and their environment* PDF for more information.
8. Create a JavaMail session. See the *Administering applications and their environment* PDF for more information.

9. Create resources for session support. See the *Administering applications and their environment* PDF for more information.
10. Configure a Session Manager. See the *Administering applications and their environment* PDF for more information.

Java memory tuning tips

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

In particular, verify the following:
- The application is not over utilizing objects
- The application is not leaking objects
- The Java heap parameters are set properly to handle a given object usage pattern

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

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

The garbage collection bottleneck

Examining Java garbage collection gives insight to how the application is utilizing memory. Garbage collection is a Java strength. By taking the burden of memory management away from the application writer, Java applications are more robust than applications written in languages that do not provide garbage collection. This robustness applies as long as the application is not abusing objects. Garbage collection normally consumes from 5% to 20% of total execution time of a properly functioning application. If not managed, garbage collection is one of the biggest bottlenecks for an application.

The i5/OS JVM uses concurrent (asynchronous) garbage collection. This type of garbage collection results in shorter pause times and allows application threads to continue processing requests during the garbage collection cycle.

The heap size settings control garbage collection in the i5/OS JVM. The initial heap size is a threshold that triggers new garbage collection cycles. For example, if the initial heap size is 10 MB, a new collection cycle is triggered as soon as the JVM detects that since the last collection cycle, 10 MB are allocated.

Smaller heap sizes result in more frequent garbage collections than larger heap sizes. If the maximum heap size is reached, the garbage collector stops operating asynchronously, and user threads are forced to wait for collection cycles to complete. This situation has a significant negative impact on performance. A maximum heap size of 0 (*NOMAX) assures that garbage collection always operates asynchonously. For more information about tuning garbage collection with the JVM heap settings, see the *Tuning guide* PDF.

Monitoring garbage collection

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

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.

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

For more information about monitoring garbage collection, see:

- The description of the DMPJVM command in the i5/OS Information Center. This command dumps JVM information for a specific job.
- The iDoctor for iSeries for a description of the Heap Analysis Tools for Java. This tool is a component of the iDoctor for iSeries suite of performance monitoring tools. The Heap Analysis Tools component performs Java application heap analysis and object create profiling (size and identification) over time. This tool is sometimes called Java Watcher or Heap Analyzer.
- The description of Performance Explorer (PEX) contained in the i5/OS Information Center topic ″Tuning Garbage Collection for Java(TM) and WebSphere on iSeries.″ You can use a Performance Explorer (PEX) trace to determine how much CPU is being used by the garbage collector.

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. See the *Tuning guide* PDF for more information about JVMPI counters.

You can also use the following tools to monitor JVM object creation:

- The DMPJVM command. The DMPJVM (Dump Java Virtual Machine) command dumps JVM information for a specific job.
- The ANZJVM command. The ANZJVM (Analyze Java Virtual Machine) command collects information about the Java Virtual Machine (JVM) for a specified job. This command is available in i5/OS V5R2 and higher.
- The Performance Trace Data Visualizer (PTDV)

The best result for the average time between garbage collections is at least 5-6 times the average duration of a single garbage collection. If you do not achieve this number, the application is spending more than 15% of its time in garbage collection.

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

Detecting memory leaks

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

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

Memory leak testing

The goal of memory leak testing is to magnify numbers. Memory leaks are measured in terms of the amount of bytes or kilobytes that cannot be garbage collected. The delicate task is to differentiate these amounts between expected sizes of useful and unusable memory. This task is achieved more easily if the

numbers are magnified, resulting in larger gaps and easier identification of inconsistencies. The following list contains important conclusions about memory leaks:

- **Long-running test**

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

- **Repetitive test**

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

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

- **Concurrency test**

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

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

You can use these tools to detect memory leaks:

- Tivoli Performance Viewer. For more information and examples of using the Tivoli Performance Viewer to detect memory leaks, see the topic ″Tuning Garbage Collection for Java(TM) and WebSphere on iSeries.″ in the i5/OS Information Center.
- The DMPJVM command. The DMPJVM (Dump Java Virtual Machine) command dumps JVM information for a specific job.
- The ANZJVM command. The ANZJVM (Analyze Java Virtual Machine) command collects information about the Java Virtual Machine (JVM) for a specified job. This command is available in i5/OS V5R2 and higher.
- The Heap Analysis Tools for Java. This tool is a component of the iDoctor for iSeries suite of performance monitoring tools. The Heap Analysis Tools component performs Java application heap analysis and object create profiling (size and identification) over time. This tool is sometimes called Java Watcher or Heap Analyzer.

For the best results, repeat experiments with increasing duration, like 1000, 2000, and 4000 page requests. The Tivoli Performance Viewer graph of used memory should have a sawtooth shape. Each drop on the graph corresponds to a garbage collection. There is a memory leak if one of the following occurs:

- The amount of memory used immediately after each garbage collection increases significantly. The sawtooth pattern looks more like a staircase.
- The sawtooth pattern has an irregular shape.

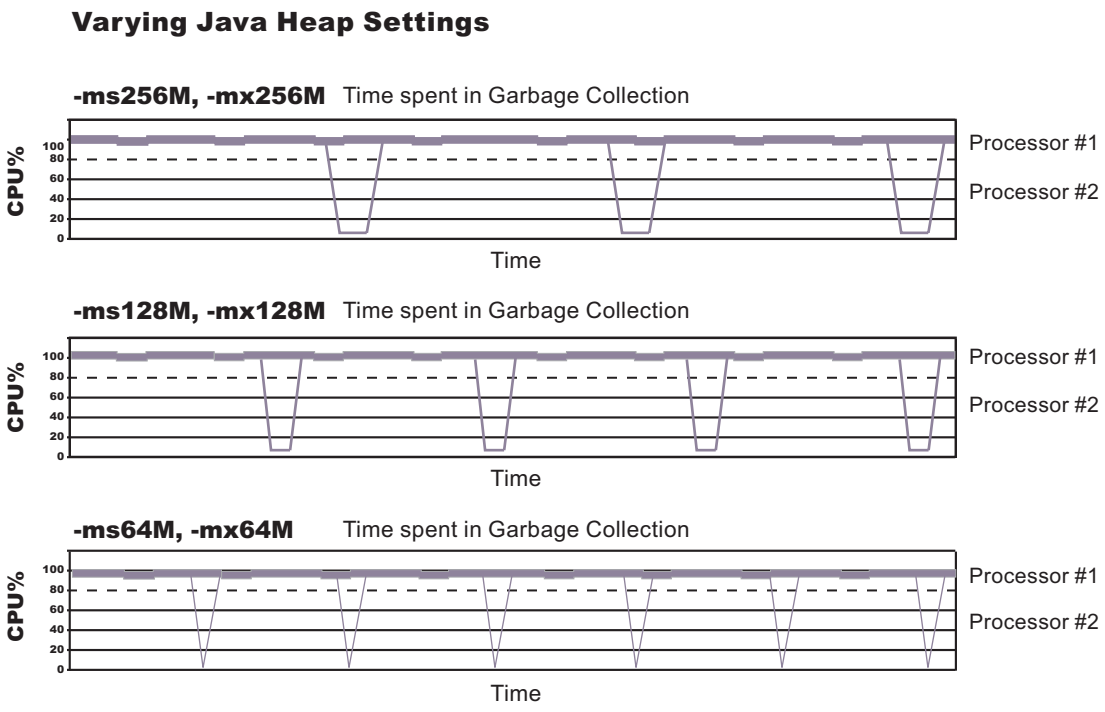
Also, look at the difference between the number of objects allocated and the number of objects freed. If the gap between the two increases over time, there is a memory leak.

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

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

Initial heap size

When tuning a production system where the working set size of the Java application is not understood, it is recommended that you set the initial heap size to 96MB per processor. The total heap size in an i5/OS JVM can be approximated as the sum of the amount of live (in use) heap space at the end of the last garbage collection plus the initial heap size.



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. 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.

Important: Unlike other JVM implementations, a large amount of heap free space is not generally a concern for the i5/OS JVM.

Maximum heap size

The maximum heap size can affect application performance. The maximum heap size specifies the maximum amount of object space the garbage collected heap can consume. If the maximum heap size is too small, performance might degrade significantly, or the application might receive out of memory errors when the maximum heap size is reached.

Because of the complexity of determining a correct value for the maximum heap size, a value of 0 (meaning there is no size limit) is recommended unless an absolute limit on the object space for the garbage collected heap size is required.

If you want to determine the proper value for the maximum heap size, you must run multiple tests, because the appropriate value is different for each configuration or workload combination. If you want to prevent a run-away JVM, set the maximum heap size larger than you expect the heap to grow, but not so large that it affects the performance of the rest of the machine.

For one of the tests you should:

1. Run your application server under a heavy load with a maximum heap value of 0
2. Use the DMPJVM command or iDoctor to determine the maximum size of the garbage collected heap for the JVM.
3. Multiply the size of the garbage collection heap by 1.25. The result is a reasonable estimate for maximum heap size becasue the smallest acceptable value for the maximum heap size is 125 percent of the garbage collected heap size.

Because you can specify a larger value for the maximum heap size without affecting performance, it is recommended that you set the largest possible value based on the resource restrictions of the JVM or the limitations of your system configuration.

After you determine an appropriate value for the maximum heap size, you might need to set up or adjust the pool in which the JVM runs. By default, WebSphere Application Server jobs run in the base system pool (storage pool 2 as shown by WRKSYSSTS), but you can specify a different pool. The maximum heap size should not be set larger than 125 percent of the size of the pool in which the JVM is running. It is recommended that you run the JVM in its own memory pool with the memory permanently assigned to that pool, if possible.

If the performance adjuster is set to adjust the memory pools (that is, the system value QPFRADJ is set to a value other than 0), it is recommended that you specify a minimum size for the pool using

WRKSHRPOOL. The minimum size should be approximately equal to your garbage collected heap working set size. Setting a correct maximum heap size and properly configuring the memory pool can prevent a JVM with a memory leak from consuming system resources, but still offers excellent performance.

When a JVM must run in a shared pool, it is more difficult to determine an appropriate value for the maximum heap size. Other jobs running in the pool can cause the garbage collected heap pages to be aged out of the pool. If the garbage collected heap pages are aged out of the pool, the garbage collector must fault the pages back into the pool on the next garbage collection cycle because it needs to access all of the pages in the garbage collected heap. Because the i5/OS JVM does not stop all of the JVM threads to clean the heap, excessive page faulting causes the garbage collector to slow down and the garbage collected heap to grow. Instead the size of the heap is increased, and threads continue to run.

This heap growth is an artificial inflation of the garbage collected heap working set size, and must be considered if you want to specify a maximum heap value. When a small amount of artificial inflation occurs, the garbage collector reduces the size of the heap over time if the space remains unused and the activity in the pool returns to a steady state. However, in a shared pool, you might experience problems if the maximum heap size is not set correctly:

- If the maximum heap size is too small, artificial inflation can result in severe performance degradation or system failure if the JVM throws an out-of-memory error.
- If the maximum heap size is set too large, the garbage collector might reach a point where it is unable to recover the artificial inflation of the garbage collected heap. In this case, performance is also negatively affected. A value that is too large might also keep the garbage collector from preventing a JVM failure. However, the garbage collector is still able to prevent a run-away JVM from consuming excessive amounts of system resources.

If you must set the maximum heap size to guarantee that the heap size does not exceed a given level, specify an initial heap size that is 80-90% smaller than the maximum heap size. However, the value specified should be large enough to not negatively affect performance.

Configuration update performance in a large cell configuration

In a large cell configuration, you might have to determine whether configuration update performance or consistency checking is more important. When configuration consistency checking is turned on, a large amount of time might be required to save a configuration change or to deploy a large number of applications. The following factors influence how much time is required:

- The more application servers or clusters there are defined in cell, the longer it takes to save a configuration change.
- The more applications there are deployed in a cell, the longer it takes to save a configuration change.

If the amount of time required to change a configuration change is unsatisfactory, you can add the config_consistency_check custom property to your JVM settings and set the value of this property to false.

1. In the administrative console, click **System administration > Deployment manager**.
2. Under Server Infrastructure, select Java and Process Management, and then click **Process Definition**.
3. Under Additional Properties, click **Java Virtual Machine > Custom Properties > New**.
4. Enter config_consistency_check in the Name field and false in the Value field.
5. Click **OK** and then **Save** to apply these changes to the master configuration.
6. Restart the server.

Configuring multiple network interface support

Application servers, by default, are configured to use all of the network interfaces that are available for them to use. You can change this configuration such that an application server only uses a specific network interface. However, you cannot configure it to use a subgroup of interfaces. For example, if you have three ethernet adapters, you cannot configure an application server to use two of the three adapters.

When an application server is configured to use all network interfaces, if it opens a socket on port 9901 on a machine with two TCP/IP addresses, it opens port 9901 on both IP addresses.

When an application server is configured to use a specific network interface, it only communicates on that one network interface. For example, on a Windows operating system, if an application server opens a socket on port 7842 on an ethernet adapter with an address of 192.168.1.150, the netstat output displays 192.168.1.150.7842 in the Local Address field, indicating that port 7842 is only bound to 192.168.1.150.

If you have more than one network interface and you want to use each one separately, you must have a separate configuration profile for each interface. When network interfaces are used separately, a separate node agent is required for each network interface that has an application server running on it. Two application servers bound to two separate network interfaces on the same machine cannot be in the same node because they have different TCP/IP addresses.

Important:

- If you want a specific application server to use a single network interface, perform the following steps for that application server.
- If you want an entire node to use a single network interface, perform the following steps for your node agent and all the application servers in that node.
- If you want an entire cell to use a single network interface, perform the following steps for the deployment manager, node agent, and all the application servers in the node.
- When performing the following steps, do not specify localhost, a loop back address, such as 127.0.0.1, or an * (asterisk) for the TCP/IP addresses.

1. Update the com.ibm.CORBA.LocalHost and com.ibm.ws.orb.transport.useMultiHome Object Request Broker (ORB) custom properties.
   a. In the administrative console, navigate to the indicated page.
      - For an application server, click **Servers** > **Application Servers** > *server*. Then, under Container Settings, click **Container services** > **ORB Service** and, under Additional properties, click **Custom Properties**.
      - For a deployment manager, click **System administration** > **Deployment manager**, and under Additional Properties, click **ORB Service**. Then, under Additional properties, click **Custom Properties**.
      - For a node agent, click **System administration** >**Node agent** > *nodeagent* , and under Additional Properties, click **ORB Service**. Then, under Additional properties, click **Custom Properties**.

   b. Select the com.ibm.CORBA.LocalHost custom property and specify an IP address or hostname in the Value field. Do not set this property to either localhost or *.

      If the com.ibm.CORBA.LocalHost property is not in the list of already defined custom properties, click **New** and then enter `com.ibm.CORBA.LocalHost` in the Name field and specify an IP address or hostname in the Value field.

   c. Select the com.ibm.ws.orb.transport.useMultiHome custom property and specify `false` in the Value field. If the com.ibm.ws.orb.transport.useMultiHome property is not in the list of already defined custom properties, click **New** and then enter `com.ibm.ws.orb.transport.useMultiHome` in the Name field and specify `false` in the Value field.
2. Update the Java virtual machine (JVM) com.ibm.websphere.network.useMultiHome custom property for discovery and SOAP connections.

a. In the administrative console, navigate to the indicated page.
   - For an application server, click **Servers > Application Servers >** *server* **> Java and Process Management > Process Definition > > Java Virtual Machine > Custom Properties**.
   - For a deployment manager, click **System administration > Deployment manager > Java and Process Management > Process definition > Java Virtual Machine > Custom Properties**.
   - For a node agent, click **System administration >Node agent >** *nodeagent* **> Java and Process Management > Process definition > Java Virtual Machine > Custom Properties**.

b. Select the com.ibm.websphere.network.useMultiHome custom property and specify `false` in the Value field. If the com.ibm.websphere.network.useMultiHome property is not in the list of already defined custom properties, click **New** and then enter `com.ibm.websphere.network.useMultiHome` in the Name field and specify `false` in the Value field.

3. Update the host name for TCP/IP connections.
   a. In the administrative console, navigate to the indicated page.
      - For an application server, click **Servers > Application Servers >** *server*, and then, under Communications, click **Ports**.
      - For a deployment manager, click **System administration > Deployment manager**, and then under Additional Properties, click **Ports**.
      - For a node agent, click **System administration >Node agent >** *nodeagent* , and then, under Additional Properties, click **Ports**.

   b. Update the Host field for each of the listed ports to the value specified for the com.ibm.CORBA.LocalHost ORB custom property in the first step. When you finish, none of the entries listed in the Host column should contain an * (asterisk).
4. Change the Initial State setting for each of the JMS servers to Stopped .
   a. In the administrative console, click **Servers > JMS Servers.**
   b. Click one of the listed JMS servers and change the value specified for the Initial State field to Stopped.
   c. Repeat the previous step until the Initial State setting for all of the listed JMS servers is Stopped.
5. Change the Initial State setting for each of the listener ports to Stopped .
   a. In the administrative console, click **Servers > Application servers >** *server*.
   b. Under Communications, click **Messaging > Message Listener Service > Listener Ports**
   c. Click one of the listed listener ports and change the value specified for the Initial State field to `Stopped`.
   d. Repeat the previous step until the Initial State setting for all of the listed listener ports is `Stopped`.
6. Save the updates and synchronize the changes with all of the node agents in the cell.
   a. In the administrative console, click **System administration > Save Changes to Master Repository**.
   b. Select Synchronize changes with nodes, and then click **Save**.
7. Stop and restart all the affected servers, node agents, and the deployment manager.

You have configured an installation of WebSphere Application Server to communicate on one, and only one network interface on a machine that has more than one network interface.

This example creates two nodes, each using a separate network interface, on a machine that has at least two network interfaces.

1. Use the Profile Management tool to create an application server and federate it into the desired cell.
2. Use the Profile Management tool to create an application server profile, specifying a host name that is different than the host name used for the previously created application server. Federate this application server into the desired cell.

3. Start the node agent and application server that are configured to the first network interface. Follow the preceding steps for the node agent and application server to prepare this node to communicate on the network interface you specified when you configured this application server.
4. Start the second node agent and application server. Follow the preceding steps for the node agent and application server to prepare this node to communicate only on the network interface that you specified when you configured the second application server.
5. Stop all of the node agents and application servers that you created in this example.
6. Restart all of these node agents and application servers.

You have two separate nodes running on two different network interfaces.

If you are using a standalone Java client or server to communicate with WebSphere Application Server, and you are using the WebSphere Application Server Software Development Kit (SDK), add the following properties to your Java command to enable the ORB for your application to communicate with a specific network interface.

```
-Dcom.ibm.ws.orb.transport.useMultiHome=false
-Dcom.ibm.CORBA.LocalHost=host_name
```

host_name is the TCP/IP address or *hostname* of the network interface for the ORB to use.

Important: Do not set *host_name* to localhost, a loop back address, such as 127.0.0.1, or an * (asterisk).

Tuning application servers

The WebSphere Application Server contains interrelated components that must be harmoniously tuned to support the custom needs of your end-to-end e-business application.

This group of interrelated components is known as the queuing network. The queuing network helps the system achieve maximum throughput while maintaining the overall stability of the system.

The following steps describe various tuning tasks that may improve your application server performance. You can choose to implement any of these application server settings. These steps can be performed in any order.

1. **Tune the object request broker.** An Object Request Broker (ORB) manages the interaction between clients and servers, using the Internet InterORB Protocol (IIOP). It supports client requests and responses received from servers in a network-distributed environment. You can use the following parameters to tune the ORB:
   - Set **Pass by reference (com.ibm.CORBA.iiop.noLocalCopies)** as described in the *Tuning guide* PDF.
   - Set the **Connection cache minimum (com.ibm.CORBA.MaxOpenConnections)** as described in the *Tuning guide* PDF.
   - Set **Maximum size** as described in Thread pool settings
   - Set **com.ibm.CORBA.ServerSocketQueueDepth** as described in the *Administering applications and their environment* PDF.
   - Set the **com.ibm.CORBA.FragmentSize** as described in the *Administering applications and their environment* PDF.
2. **Tune the XML parser definitions.**
   - **Description:** Facilitates server startup by adding XML parser definitions to the jaxp.properties and xerxes.properties files in the ${*app_server_root*}/jre/lib directory. The XMLParserConfiguration value might change as new versions of Xerces are provided.
   - **How to view or set:** Insert the following lines in both files:

```
javax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl
javax.xml.parsers.DocumentBuildFactory=org.apache.xerces.jaxp.
          DocumentBuilderFactoryImpl
org.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.
         StandardParserConfiguration
```

- **Default value:** None
- **Recommended value:** None

3. **Tune the dynamic cache service.**

   Using the dynamic cache service can improve performance. See the *Administering applications and their environment* PDF for information about using the dynamic cache service and how it can affect your application server performance.

4. **Tune the Web container.** The WebSphere Application Server Web container manages all HTTP requests to servlets, JavaServer Pages and Web services. Requests flow through a transport chain to the Web container. The transport chain defines the important tuning parameters for performance for the Web container. There is a transport chain for each TCP port that WebSphere Application Server is listening on for HTTP requests. For example, the default HTTP port 9080 is defined in Web container inbound channel chain. Use the following parameters to tune the Web container:
   - HTTP requests are processed by a pool of server threads. The minimum and maximum thread pool size for the Web container can be configured for optimal performance. Generally, 5 to 10 threads per server CPU will provide the best throughput. The number of threads configured does not represent the number of requests WebSphere can process concurrently. Requests are queued in the transport chain when all threads are busy. To specify the thread pool settings:
     a. Click **Servers > Application Servers >** *server_name* **Web Container Settings> Web Container > Web container transport chains**.
     b. Select the normal inbound chain for serving requests. This will usually be named WCInboundDefault, on port 9080.
     c. Click **TCP Inbound Channel (TCP_2)**.
     d. Set **Thread Pools** under Related Items.
     e. Select **WebContainer**.
     f. Enter values for **Minimum Size** and **Maximum Size**.
   - The HTTP 1.1 protocol provides a keep-alive feature to enable the TCP connection between HTTP clients and the server to remain open between requests. By default WebSphere Application Server will close a given client connection after a number of requests or a timeout period. After a connection is closed, it will be recreated if the client issues another request. Early closure of connections can reduce performance. Enter a value for the maximum number of persistent requests to (keep-alive) to specify the number of requests that are allowed on a single HTTP connection. Enter a value for persistent timeouts to specify the amount of time, in seconds, that the HTTP transport channel allows a socket to remain idle between requests. To specify values for Maximum persistent requests and Persistent timeout:
     a. Click **Servers > Application Servers >** *server_name* **Web Container Settings> Web Container > Web container transport chains**.
     b. Select the normal inbound chain for serving requests. This will usually be named WCInboundDefault, on port 9080.
     c. Click **HTTP Inbound Channel (HTTP_2)**.
     d. Enter values for **Maximum persistent requests** and **Persistent timeout**.

5. **Tune the EJB container.** An Enterprise JavaBeans (EJB) container is automatically created when you create an application server. After the EJB container is deployed, you can use the following parameters to make adjustments that improve performance.
   - Set the **Cleanup interval** and the **Cache size** as described in the *Administering applications and their environment* PDF.
   - **Break CMP enterprise beans into several enterprise bean modules** while assembling EJB modules.

   See also the *Tuning guide* PDF.

6. **Tune the session management.**

The installed default settings for session management are optimal for performance. See the *Tuning guide* PDF for more information about tuning session management.

7. **Tune the data sources and associated connection pools.** A data source is used to access data from the database; it is associated with a pool of connections to that database.
   - Review information on connection pools, contained in the *Administering applications and their environment* PDF, to understand how the number of physical connections within a connection pool can change performance.
8. **Tune the URL invocation cache.**

   Each JavaServer Page is a unique URL. If you have more than 50 unique URLs that are actively being used, increase the value specified for the invocationCacheSize JVM custom property. This property controls the size of the URL invocation cache. See the *Administering applications and their environment* PDF for more information on how to change this property.

Web services client to Web container optimized communication

To improve performance, there is an optimized communication path between a Web services client application and a Web container that are located in the same application server process. Requests from the Web services client that are normally sent to the Web container using a network connection are delivered directly to the Web container using an optimized local path. The local path is available because the Web services client application and the Web container are running in the same process.

This direct communication eliminates the need for clients and web containers that are in the same process to communicate over the network. For example, a Web services client might be running in an application server. Instead of accessing the network to communicate with the Web container, the Web services client can communicate with the Web container using the optimized local path. This optimized local path improves the performance of the application server by allowing Web services clients and Web containers to communicate without using network transports.

In a clustered environment, there is typically an HTTP server (such as IBM HTTP server) that handles incoming client requests, distributing them to the correct application server in the cluster. The HTTP server uses information about the requested application and the defined virtual hosts to determine which application server receives the request. The Web services client also uses the defined virtual host information to determine whether the request can be served by the local Web container. You must define unique values for the host and port on each application server. You cannot define the values of host and port as wild cards denoted by the asterisk symbol (*) when you enable the optimized communication between the Web services application and the Web container. Using wild cards indicate that the local Web container can handle Web services requests for all destinations.

The optimized local communication path is disabled by default. You can enable the local communication path with the enableInProcessConnections custom property. Before configuring this custom property, make sure that you are not using wild cards for host names in your Web container end points. Set this property to **true** in the Web container to enabled the optimized local communication path. When disabled, the Web services client and the Web container communicate using network transports.

For information about how to configure the enableInProcessConnections custom property, see the *Administering applications and their environment* PDF.

When the optimized local communication path is enabled, logging of requests through the local path uses the same log attributes as the network channel chain for the Web container. To use a different log file for in process requests than the log file for network requests, use a custom property on the HTTP Inbound Channel in the transport chain. Use the inProcessLogFilenamePrefix custom property to specify a string that is added to the beginning of the network log file name to create a file name that is unique. Requests through the local process path are logged to this specified file. For example, if the log filename is

../httpaccess.log for a network chain, and the inProcesslLogFilenamePrefix custom property is set to “local” on the HTTP channel in that transport chain, the local log file name for requests to the host associated with that chain is /localhttpaccess.log.

Application servers: Resources for learning

Use the following links to find relevant supplemental information about configuring application servers. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

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

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

Programming instructions and examples
- WebSphere Application Server education

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

Administration
- Listing of all IBM WebSphere Application Server Redbooks

Chapter 8. Balancing workloads with clusters

You should use server clusters and cluster members to monitor and manage the workloads of application servers.

Consider your options for configuring application servers. See “Managing application servers” on page 211 for more information.

To assist you in understanding how to configure and use clusters for workload management, consider 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.)

1. Decide which application server you want to cluster.
2. Decide whether you want to replicate data. Replication is a service that transfers data, objects, or events among application servers. See “Replicating data across application servers in a cluster” on page 327 for more information. You can create a replication domain when creating a cluster.
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. See “Creating clusters” on page 309 for more information.
5. You can create one or more cluster members of the cluster.
6. Configure a backup cluster that handles requests if the primary cluster fails.
7. Start all of the application servers by starting the cluster. Workload management automatically begins when you start the cluster members of the application server.
8. Once you have the cluster running, you can perform the following tasks:
   - Stop the cluster.
   - Upgrade applications on clusters. See the *Administering applications and their environment* PDF for more information.
   - Detect and handle problems with server clusters and their workloads.
   - Tune the behavior of the workload management run time. If your application is experiencing problems with 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.

For stand-alone Java clients, you must define a bootstrap host. 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.

Clusters and workload management

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

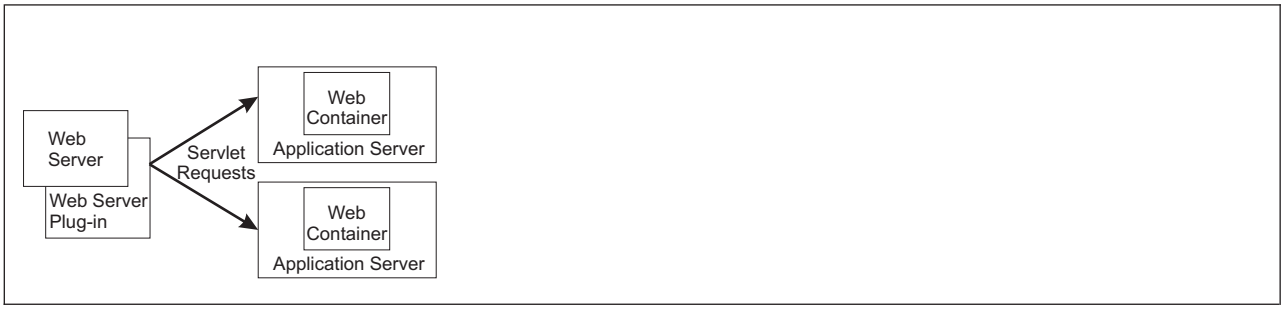
Servers that belong to a cluster are *members* of that cluster set and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do

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

When you create a cluster, you make copies of an existing application server template. The template is most likely an application server that you have previously configured. You are offered the option of making that server a member of the cluster. However, it is recommended that you keep the server available only as a template, because the only way to remove a cluster member is to delete the application server. When you delete a cluster, you also delete any application servers that were members of that cluster. There is no way to preserve any member of a cluster. Keeping the original template intact allows you to reuse the template if you need to rebuild the configuration.

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

Clustering application servers that host Web containers automatically enables plug-in workload management for the application servers and the servlets they host. The routing of servlet requests occurs between the Web server plug-in and clustered application servers using HTTP transports, or HTTP transport channels.



This routing is based on weights associated with the cluster members. If all cluster members have identical weights, the plug-in sends equal requests to all members of the cluster, assuming there are no strong affinity configurations. If the weights are scaled in the range from zero to twenty, the plug-in usually routes requests to those cluster members with the higher weight values.

Assign a weight to a cluster member based on its approximate, proportional ability to do work. The weight value specified for a specific member is only meaningful in the context of the weights you specify for the other members within a cluster. The weight values do not indicate absolute capability. If a cluster member is unavailable, the Web server plug-in temporarily routes requests around that cluster member.

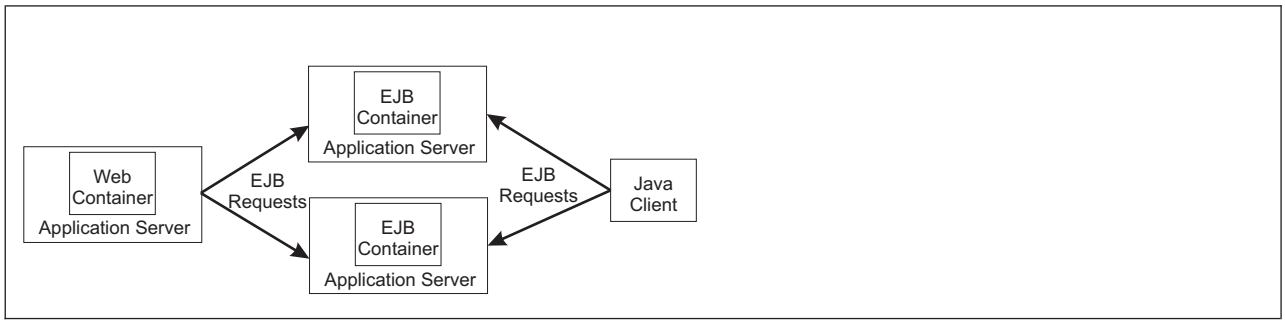
For example, if you have a cluster that consists of two members, assigning weights of 1 and 2 causes the first member to get approximately 1/3 of the workload and the second member to get approximately 2/3 of the workload. However, if you add a third member to the cluster, and assign the new member a weight of 1, approximately 1/4 of the workload now goes to the first member, approximately 1/2 of the workload goes to the second member, and approximately 1/4 of the workload goes to the third member. If the first cluster member becomes unavailable, the second member gets approximately 2/3 of the workload and third member gets approximately 1/3 of the workload.

The weight values only approximate your load balance objectives. There are other application dependencies, such as thread concurrency, local setting preferences, affinity, and resource availability that

are also factors in determining where a specific request is sent. Therefore, do not use the exact pattern of requests to determine the weight assignment for specific cluster members.

See “Cluster member settings” on page 320 for information on how to set the weight for a cluster member.

Workload management for EJB containers can be performed by configuring the Web container and EJB containers on separate application servers. Multiple application servers can be clustered with the EJB containers, enabling the distribution of enterprise bean requests between EJB containers on different application servers.



In this configuration, EJB client requests are routed to available EJB containers in a round robin fashion based on assigned server weights. The EJB clients can be servlets operating within a Web container, stand-alone Java programs using RMI/IIOP, or other EJBs.

The server weighted round robin routing policy ensures a balanced routing distribution based on the set of server weights that have been assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, the expected distribution for the cluster is that all servers receive the same number of requests. If the weights for the servers are not equal, the distribution mechanism sends more requests to the higher weight value servers than the lower weight value servers. The policy ensures the desired distribution, based on the weights assigned to the cluster members.

You can choose to have requests sent to the node on which the client resides as the preferred routing. In this case, only cluster members on that node are chosen (using the round robin weight method). Cluster members on remote nodes are chosen only if a local server is not available

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.

Multiple servers that can service the same client request form the basis for failover support. If a server fails while processing a client request, the failed request can be rerouted to any of the remaining cluster members. In fact, several servers could fail, and as long as at least one cluster member is running, client requests can continue to be serviced. For further information backing up failed processes, see “Replicating data across application servers in a cluster” on page 327.

See “Backup clusters” on page 323 for information on backup cluster support. A backup cluster continues functioning when all cluster members of the primary cluster are not available.

Clusters and node groups

Node groups bound clusters. All cluster members of a given cluster must be members of the same node group.

Any application you install to a cluster must be able to execute on any application server that is a member of that cluster. For the application you deploy to run successfully, all the members of the cluster must be located on nodes that meet the requirements for that application.

In a cell that has many different server configurations, it might be difficult to determine which nodes have the capabilities to host your application. A *node group* can be used to define groups of nodes that have enough in common to host members of a given cluster. All cluster members in a cluster must be in the same node group.

All nodes are members of at least one node group. When you create a cluster, the first application server you add to the cluster defines the node group that bounds the cluster. All other cluster members you add to the cluster can only be on nodes that are members of this same node group. When you create a new cluster member in the administrative console, you are allowed to create the application server on a node that is a member of the node group for that cluster only.

Nodes can be members of multiple node groups. If the first cluster member you add to a cluster has multiple node groups defined, the system automatically chooses the node group that bounds the cluster. You can change the node group by modifying the cluster settings. Use the “Server cluster settings” on page 315 page to change the node group.

Workload management (WLM) for all platforms except z/OS

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

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

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

Techniques for managing state

Multiple machine 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 if 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 rely on information from earlier requests. 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. 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.

Creating clusters

A cluster is a set of application servers that you manage together as a way to balance workload.

Before you create a cluster determine:

- Whether you want enterprise bean requests routed to the node on which the client resides.
- If you want to use HTTP memory-to-memory replication.
- The configuration settings you want to specify for the first cluster member. A copy of the first cluster member that you create is stored as part of the cluster data and becomes the template for all additional cluster members that you create.
- The node on which the first cluster member resides.

Also review the content of “Clusters and workload management” on page 305, especially the information about setting cluster weights.

You might want to create a cluster if you need to:

- Balance your client requests across multiple application servers.
- Provide a highly available environment for your applications.

A cluster enables you to manage a group of application servers as a single unit, and distribute client requests among the application servers that are members of the cluster.

To create a cluster:

1. In the administrative console, click **Servers > Clusters > New**. The Create a new cluster wizard starts.
2. Specify a name for the cluster.
3. Select **Prefer local** if you want to enable node-scoped routing optimization. This option is enabled by default. When this option is enabled, if possible, EJB requests are routed to the client node. This option improves performance because client requests are sent to local enterprise beans.
4. Select **Configure HTTP session memory-to-memory replication** if you want a memory-to-memory replication domain created for this cluster. The replication domain is given the same name as the cluster and is configured with the default settings for a replication domain. When the default settings

are in effect, a single replica is created for each piece of data and encryption is disabled. Also, the Web container for each cluster member is configured for memory-to-memory replication.

To change these settings for the replication domain, click **Environment > Replication domains >** *replication_domain_name*. To modify the Web container settings, click **Servers > Clusters >** *cluster_name* **> Cluster members >** *cluster_member_name* **> Web container settings > Session management > Distributed environment settings** in the administrative console. If you change these settings for one cluster member, you might also need to change them for the other members of this cluster.

5. Click **Next**.
6. Choose whether to create an empty cluster or to create the first member of the cluster.

   If you decide to create an empty cluster, when you are ready to add members to this cluster, in the administrative console, click **Servers > Clusters >** *cluster_name* **> Cluster members > New**.

   To create an empty cluster:

   a. Select **None. Create an empty cluster.**
   b. Click **Next** to display a summary of the defined cluster.
   c. Click **Finish** to create the cluster, or click **Cancel** if you decide not to create this cluster.

   When you create the first cluster member, remember that a copy of the first cluster member that you create is stored as part of the cluster data and becomes the template for all additional cluster members that you create.

   a. Specify the name of the first cluster member.
   b. Select the node on which you want this cluster member to reside.
   c. Specify the weight value for the cluster member. The weight value controls the amount of work that is directed to the application server. If the weight value for this server is greater than the weight values that are assigned to other servers in the cluster, then this server receives a larger share of the workload. The weight value represents a relative proportion of the workload that is assigned to a particular application server. The value can range from `0` to `20`. See “Clusters and workload management” on page 305 for more information.
   d. Select **Generate unique HTTP ports** if you want to generate unique port numbers for every HTTP transport that is defined in the source server. When this option is selected, which is the default setting, this cluster member does not have HTTP transports or HTTP transport channels that conflict with any of the other servers that are defined on the same node. If you unselect this option, all of the cluster members will share the same HTTP ports.
   e. Select the core group to which you want this cluster member to belong. You are prompted for the core group only if you have more than one core group defined for this cluster.
   f. Select one of the following options as the basis for the first cluster member.
      - Create the member using an application server template.
      - Create the member using an existing application server as a template.
      - Create the member by converting an existing application server.

      Important: You can only add an existing application server to the cluster if you select that server as the first cluster member. You cannot add other existing application servers to that cluster after you create the first cluster member. If you add an existing server to a cluster, the only way to remove that server from the cluster is to delete the server. Therefore, you might want to use the existing server as a template for the first cluster member instead of as the cluster member. If you keep the original application server out of the cluster, you can reuse that server as the template if you need to rebuild the configuration.
7. Click **Next**.

8. Create additional cluster members. Before you create additional cluster members, check the configuration settings of the first cluster member. These settings are displayed at the bottom of the Create additional cluster members panel of the Create a new cluster wizard. For each additional member that you want to create:
   a. Specify a unique name for the member. The name must be unique within the node.
   b. Select the node to which you want to assign the cluster member.
   c. Specify the weight you want given to this member. The weight value controls the amount of work that is directed to the application server. If the weight value for the server is greater than the weight values that are assigned to other servers in the cluster, then the server receives a larger share of the workload. The value can range from `0` to `20`.
   d. Select **Generate unique HTTP ports** if you want to generate unique port numbers for every HTTP transport that is defined in the source server.
   e. Click **Apply**. You can edit the configuration settings of any of the newly created cluster members other than the first cluster member, or you can create additional cluster members. Click **Previous** to edit the properties of the first cluster member.
9. When you finish creating cluster members, click **Next**.
10. View the summary of the cluster and then click **Finish** to create the cluster, click **Previous** to return to the previous wizard panel and change the cluster, or click **Cancel** to exit the wizard without creating the cluster.
11. To further configure a cluster, click **Servers > Clusters**, and then click the name of the cluster. Only the **Configuration** and **Local Topology** tabs appear until you save your changes.
12. Click **Review** to review your cluster configuration settings. Repeat the previous step if you need to make additional configuration changes.
13. If you do not want to make any additional configuration changes, select Synchronize changes with Nodes and then click **Save**. Your changes are saved and synchronized across all of your nodes.

    Important: If you click **Save**, but do not select Synchronize changes with Nodes, when you restart the cluster, WebSphere Application Server does not start the cluster servers because it cannot find them on the node. If you want to always synchronize your configuration changes across your nodes, you can select Synchronize changes with Nodes as one of your console preferences.
14. Restart the cluster.

You have a configured cluster to which you can assign work requests. The **Runtime** and **Local Topology** tabs appear the next time you access this page.

You can:

- Click **Servers > Clusters >** *cluster_name* to view or to change the configuration settings for a cluster. For example, if you are running in a high availability environment, you might want to select the **Enable failover of transaction log recovery** option for this cluster. This option allows the recovery of transactions to failover from one cluster member to another. See Chapter 8, “Balancing workloads with clusters,” on page 305 for more information about cluster configuration options.
- Create additional cluster members.
- Start the cluster.
- Use scripting to automate the task of creating clusters.

  See the *Administering applications and their environment* PDF for more information.
- Create a static routing table to temporarily handle IIOP routing for the cluster if your high availability infrastructure is disabled.

Creating a cluster: Basic cluster settings

Use this page to enter the basic settings for a cluster.

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

Cluster name

Specifies the name of the cluster. The cluster name must be unique within the cell.

Prefer local

Specifies that the node scoped routing optimization is enabled or disabled. The default is enabled, which means that Enterprise JavaBeans (EJB) requests are routed to the client node when possible. Enabling this setting improves performance because client requests are sent to local enterprise beans.

Configure HTTP session memory-to-memory replication

Specifies that when the cluster is created, a memory-to-memory replication domain is created for each of the members of this cluster.

If a replication domain is created, it is given the same name as the cluster and is configured with the default settings for a replication domain. When the default settings are in effect, a single replica is created for each piece of data and encryption is disabled. Also, the SIP container and Web container for each cluster member is configured for memory-to-memory replication.

To modify the replication domain settings, in the administrative console, click **Environment > Replication domains >** *replication_domain_name*.

The default mode setting for the replication domain is `Both client and server`. In this mode, all data sent to either the client or the server is replicated. This setting is good for an environment that has a middle to low traffic load. However, if your environment has a high traffic load, you should change the replication domain mode setting to either `Client only` , or `Server only`, because these settings provide better scaling. In `Client only` mode, only data sent to the client is replicated. In `Server only` mode, only data sent to the server is replicated.

To modify the mode setting for a replication domain, in the administrative console, click **Servers > Clusters >***cluster_name* **> Cluster members >** *cluster_member_name*, and then click either **SIP Container Settings** or **Web Container Settings**. Next click **Session management > Distributed environment settings**, and select a different mode. It does not matter whether you change the mode under the SIP container or the Web container because the same replication domain settings apply to both containers.

Important: If you change any of the replication domain settings for one cluster member, including the mode setting, you should change them for all of the other members of the cluster.

Creating a cluster: Create first cluster member

Use this page to specify settings for the first cluster member.

There are two ways to create the first member of a cluster:

- You can create the first member when you create a new cluster.

  To create a new cluster, in the administrative console, click **Servers > Clusters > New**.
- You can create an empty cluster and then add a first member after you finish creating the cluster.

  To create a cluster member for an existing cluster, in the administrative console, click **Servers > Clusters >** *cluster_name* **> Cluster members > New**.

When you create the first cluster member, a copy of that member is stored as part of the cluster data and becomes the template for all additional cluster members that you create.

When adding servers to a cluster, remember that the only way to remove an application server from a cluster is to delete the application server from the list of cluster members.

Member name
Specifies the name of the application server that is created for the cluster.

The member name must be unique on the selected node.

Select node
Specifies the node on which the application server resides.

Weight
Specifies the amount of work that is directed to the application server.

If the weight value for the server is greater than the weight values that are assigned to other servers in the cluster, the server receives a larger share of the cluster workload. The value can range from 0 to 20. Enter zero to indicate that you do not want requests to route to this application server unless this server is the only server that is available to receive requests.

Core group
Specifies the core group in which the application server resides. This field displays only if you have multiple core groups configured. You can change this value only for the first cluster member.

Generate unique HTTP ports
Specifies that a unique HTTP port is generated for the application server. By generating unique HTTP ports for the application server, you avoid potential port collisions and configurations that are not valid.

Select basis for first cluster member:
Specifies the basis you want to use for the first cluster member.

If you select **Create the member using an application server template**, the settings for the new application server are identical to the settings of the application server template you select from the list of available templates.

If you select **Create the member using an existing application server as a template**, the settings for the new application server are identical to the settings of the application server you select from the list of existing application servers.

If you select **Create the member by converting an existing application server**, the application server you select from the list of available application servers becomes a member of this cluster.

If you select **None. Create an empty cluster** , a new cluster is created but it does not contain any cluster members.

Important: The basis options are available only for the first cluster member. All other members of a cluster are based on the cluster member template which is created from the first cluster member.

Creating a cluster: Summary settings
Use this administrative console page to view and save settings when you create a cluster or cluster member.

You can view this administrative console page whenever you create a new cluster or a new cluster member. This summary page displays your configuration changes before you commit the changes and the new cluster or cluster member is created.

To create a cluster, in the administrative console, click **Servers > Clusters > New**.

To create a cluster member for an existing cluster, in the administrative console, click **Servers > Clusters > *cluster_name* > Cluster members > New**

The bounding node group of the cluster is based on the first application server that is added as a member of the cluster. To select a different bounding node group, click **Servers > Clusters >** *cluster_name* in the administrative console and edit the settings for that cluster.

Review the changes to your configuration, and then click **Finish** to complete and save your work.

Creating a cluster: Create additional cluster members

Use this page to create additional members for a cluster. You can add a member to a cluster when you create the cluster or after you create the cluster. A copy of the first cluster member that you create is stored as part of the cluster data and becomes the template for all additional cluster members that you create.

To add members to a cluster, in the administrative console, click **Servers > Clusters >** *cluster_name* **> Cluster members > New**. After you enter the required information about the new cluster member, click **Add Member** to add this member to the cluster member list.

After adding a cluster member, you might need to change one or more of the property settings for this cluster member, or a another cluster member that you just added. To change one or more property settings for any cluster member that you just added, other than the first cluster member, select that cluster member, and then click **Edit**. When you finish changing the property settings, click **Update Member** to save your changes.

If you decide not to create a particular cluster member, select the member and then click **Delete**.

You cannot edit or delete the first cluster member or an already existing cluster member.

If you create additional cluster members immediately after you create the first cluster member, the list of cluster members includes a checklist in front of the names of these additional cluster members. However, a check box does not appear in front of the name of the first cluster member because you cannot delete this member or edit its settings. To modify the first cluster member, click **Previous**.

Similarly, if you are adding cluster members to a cluster that already has existing members, the existing members appear in the list of cluster members but a check box does not appear in front of the names of these cluster members. To delete one of these existing members or to change the settings of one of these cluster members, in the administrative console click **Servers > Clusters >** *cluster_name* **> Cluster members** and then select the member that you want to delete or whose configuration settings you want to change.

Member name

Specifies the name of the application server that is created for the cluster.

The member name must be unique on the selected node.

Select node

Specifies the node on which the application server resides.

Weight

Specifies the amount of work that is directed to the application server.

If the weight value for the server is greater than the weight values that are assigned to other servers in the cluster, the server receives a larger share of the cluster workload. The value can range from 0 to 20. Enter zero to indicate that you do not want requests to route to this application server unless this server is the only server that is available to receive requests.

Generate unique HTTP ports
Specifies that a unique HTTP port is generated for the application server. By generating unique HTTP ports for the application server, you avoid potential port collisions and configurations that are not valid.

Server cluster collection
Use this page to view information about and change configuration settings for a cluster. A cluster consists of a group of application servers. If one of the application servers fails, requests are routed to other members of the cluster.

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

To define a new cluster, click **New** to start the Create a new cluster wizard.

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

Status
Specifies whether a cluster is stopped, partially started, started, or partially stopped.

If all cluster members are stopped, `stopped` displays for the status and state of the cluster. After you click **Start** or **Ripplestart** to start a cluster, the cluster state briefly displays as `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 `partially started`. The state remains `partially started` until all cluster members are running. When all cluster members are running, the state changes to `running` and the status is `started`. Similarly, when you click **Stop** or **ImmediateStop** to stop a cluster, the state changes to `partially stopped` when the first member stops and then changes to `stopped` when all cluster members are not running.

Server cluster settings
Use this page to view or change the configuration of a server cluster instance, and to view the local topology of a server cluster instance.

To change the configuration and local topology of a server cluster, in the administrative console click **Servers > Clusters >** *cluster_name*.

To view runtime information, such as the state of the server cluster, click **Servers > Clusters >** *cluster_name*, and then click the **Runtime** tab.

To display the topology of a specific cluster, click **Servers > Clusters >** *cluster_name*, and then click the **Local Topology** tab.

If the high availability infrastructure is disabled and you require IIOP routing capabilities, click **Servers > Clusters >** *cluster_name*, then click on the **Runtime** tab, and then click **Export route table** to take a snapshot of the run-time cluster routing information as viewed by the Deployment Manager, and serialize it to the file system under the cluster's configuration directory. A static route table for IIOP cluster traffic is created. You should then force a node synchronize to ensure that the file is distributed to all of the nodes in the cell.

Because the information contained in the static route table does not account for server run-time state, you should only use this option if the high availability infrastructure is disabled.

Use of a static route table preempts the use of the dynamic routing table that is contained in cluster members. After the static file is transferred to a node, whenever a cluster member residing in that node starts, it uses the static table instead of the dynamic table to handle IIOP routing. If a cluster member is running when you create the static route table, you must restart that cluster member to give it access to the static route table information because the table is loaded at runtime.

After the table is created, an informational message, similar to the following message, is issued that indicates the name of the file that contains the table and where that file is located:

```
The route table for cluster MyCluster was exported to file
/home/myInstall/was/server/profiles/dmgrProfile/config/cells/
   MyCell/clusters/Myfile.wsrttble.
```

As this message illustrates, the file containing the table is placed in the `config` directory of the deployment manager for this cluster. You should keep a record of this location so that you can delete this file when you are ready to start using dynamic routing again.

Important: If you use this option, you must statically set the ORB_LISTENER_ADDRESS port on each of the cluster members because the route table is static and the cluster members do not communicate during state changes. If this port is not assigned, the cluster members restart on different ports and the static routing information is not able to route requests to the cluster members.

Cluster name:

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

Data type String

Bounding node group name:

Specifies the node group that bounds this cluster. All application servers that are members of a cluster must be on nodes that are members of the same node group.

A node group is a collection of WebSphere Application Server nodes. A node is a logical grouping of managed servers, usually on a computer system that has a distinct IP host address. All application servers that are members of a cluster must be on nodes that are members of the same node group. Nodes that are organized into a node group need enough capabilities in common to ensure that clusters formed across the nodes in the node group can host the same application in each cluster member. A node must be a member of at least one node group and can be a member of more than one node group.

Create and manage node groups by clicking **System administration** > **Node groups** in the administrative console.

Enable failover of transaction log recovery:

Specifies that for the transaction service component, failover of the transaction log for recovery purposes is enabled or disabled. The default is disabled.

When this setting is enabled, and the transaction service properties required for peer recovery of failed application servers in a cluster are properly configured, failover recovery of the transaction log occurs if the server processing the transaction log fails. If the transaction services properties required for peer recovery of failed application servers in a cluster are not properly configured, this setting is ignored.

State:

Specifies whether the cluster is stopped, starting, or running.

If all cluster members are stopped, the cluster state is *stopped*. After you request to start a cluster, 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 *websphere.cluster.partial.start*. The state remains *partially started* until all cluster members are running, then the state changes to *running*. Similarly, when stopping a cluster, the state changes to *partially stopped* as the first member stops and changes to *stopped* when all members are not running.

Data type	String
Range	Valid values are starting, partially started, running, partially stopped, or stopped.

Cluster topology

Use this page to display, in a tree format, a list of all of the application server clusters defined for your WebSphere Application Server environment. The list shows all of the nodes and cluster members that are included in each cluster contained in a cell.

To view this page, in the administrative console, click **Servers > Cluster topology**.

Enabling static routing for a cluster

If your high availability infrastructure is disabled and you require IIOP routing capabilities, you can create a static routing table for the members of a cluster to use to handle enterprise bean requests. Because the information contained in this static routing table does not account for server runtime state, you should delete this table and return to using the dynamic routing table as soon as your high availability infrastructure is enabled.

Before you create a static route table, ensure that the ORB_LISTENER_ADDRESS port is set on each of the cluster members. Because the route table you create is static, and the cluster members do not communicate during state changes, if you do not set the ORB_LISTENER_ADDRESS port on each of the cluster members, the cluster members restart on different ports and the deployment manager is not able to route IIOP requests to the cluster members.

You should only create a static route table if your high availability infrastructure is disabled and you require IIOP routing capabilities. To create a static route table:

1. In the administrative console, click **Servers > Clusters**.
2. Select the fully-configured cluster for which static routing is required and click **Start** if it is not already running.
3. Ensure that the cluster is in a fully available state such that a client can route across the entire cluster.
4. Click the name of the cluster, and then click **Export route table** to create a static route table for that cluster. After the table is created, an informational message, similar to the following message, is issued that indicates the name of the file that contains the table and where that file is located:

   ```
   The route table for cluster MyCluster was exported to file
   /home/myInstall/was/server/profiles/dmgrProfile/config/cells/MyCell/
        clusters/Myfile.wsrttble.
   ```

   As this message illustrates, the file containing the table is placed in the config directory of the deployment manager for that cluster. You should keep a record of this location so that you can delete this file when you are ready to start using dynamic routing again.
5. Click **Save** , and then click **Synchronize changes with Nodes**. This step forces a node synchronization which ensures that the file is distributed to all of the nodes in the cell.
6. Select the cluster again and click **Stop** to stop the cluster.
7. Select the cluster again and click **Start**. The cluster members do not start to use the static route table until you stop and then restart the cluster.

The cluster members use the static route table to perform IIOP routes.

When your high availability infrastructure is enabled, disable static routing so that the cluster members resume using dynamic routing:

1. Set the ORB_LISTENER_ADDRESS property back to 0 (zero).
2. Delete the static route table file from the config directory of the deployment manager for this cluster.
3. Force a node synchronize to ensure the file is removed from the nodes.
4. Stop the cluster and then start the cluster again.

Adding members to a cluster

You create an application server as a member of a configured cluster.

Create a cluster if you do not already have a configured cluster.

See “Creating clusters” on page 309 for more information.

If you are migrating from a previous version of WebSphere Application Server, you can upgrade a portion of the nodes in a cell, while leaving others at the other release level. For a time, you might be managing servers that are at a previous release level and servers that are running at the newer release in the same cell. In this mixed environment, you can only add cluster members for the WebSphere Application Server versions that already exist in the cluster. For example, if a cluster contains a WebSphere Application Server v5.x member, you can create a new v5.x member in that cluster. However, if a cluster does not contain a WebSphere Application Server v5.x member, you cannot create a new v5.x member in that cluster.

To create a new cluster member, view information about existing cluster members, or manage existing cluster members:

1. In the administrative console, click **Servers > Clusters >** *cluster_name* > **Cluster members**. The Cluster members page lists members of a cluster, and for each member indicates:
   - The node on which the member resides.
   - The version of the application server. This information shows whether the cluster is a mixed cluster.
   - The configured weight for the member.
   - The runtime weight for the member. This weight indicates the proportionate workload that is currently directed to this cluster member.
   - Whether the member is started, stopped, or encountering problems.
2. Click **New** to create a new cluster member.

   Clicking **New** starts the Create a new cluster member wizard. This wizard allows you to add new members to an already configured cluster.

   A copy of the first cluster member that you create is stored as part of the cluster data and becomes the template for all additional cluster members that you create.

   a. Specify a name for the cluster member (application server). The name must be unique within the node.
   b. Select the node for the cluster member.
   c. Specify the server weight. The weight value controls the amount of work that is directed to the application server. If the weight value for the server is greater than the weight values that are assigned to other servers in the cluster, then the server receives a larger share of the workload. The value can range from `0` to `20`.
   d. Specify whether to generate unique HTTP ports.

e. Click **Add Member** to finish defining the cluster member. The first cluster member for this cluster is used as the template for this cluster member. You can repeat these steps to define other cluster members.

f. When you finish defining additional cluster members, review the summary information for the new cluster members. If you have to change any of the property settings for any of the new members, select that cluster member, and then click **Edit**. When you finish changing the property settings, click **Update Member** to save your changes.

g. When you finish defining new cluster members, click **Next** to view the summary page for the cluster, and then click **Finish** to create these new cluster members.

3. Click **Review**, select Synchronize changes with Nodes, and then click **Save** to save your changes.

You created application servers that are members of an existing server cluster.

If when you created the new members, you chose to generate unique ports, update the alias list for the virtual host that you plan to use with the new servers.

You can:

- Click on the name of the cluster member under **Member name** on the Cluster members page in the administrative console, and examine the settings for a specific cluster member.
- Click **Servers > Application Servers >** *cluster_member_name* or click **Servers > Clusters >** *cluster_name* **> Cluster members >** *cluster_member_name* to specify additional application server properties for a cluster member.
- Create a backup cluster. See “Creating backup clusters” on page 322 for more information.
- Start the cluster. See “Starting clusters” on page 326 for more information.
- Use scripting to automate the task of adding cluster members.

  See the *Using the administrative clients* PDF for more information.

Cluster member collection

Use this page to view and manage application servers that belong to a cluster.

You can also use this page to change the weight of any of the listed application servers.

A copy of the first cluster member that you create is stored as part of the cluster data and becomes the template for all additional cluster members that you create.

Any individual configuration change that you make to a cluster member does not affect the configuration settings of the cluster member template. You must use wsadmin commands to modify this template. Similarly, any changes that you make to the template does not affect existing cluster members.

See the *Using the administrative clients* PDF for more information on how to modify this template.

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

Member name

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

Node

Specifies the name of the node for the cluster member.

Version

Specifies the version of the WebSphere Application Server product on which the cluster member runs.

Configured weight
Specifies the weight that is currently configured for the cluster member. The weight determines the amount of work that is 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, the server receives a larger share of the cluster workload.

To change the configured weight for a cluster member you can either specify a new weight in the Configured weight field and click the **Update** button for the Configured weight column, or click on the name of the cluster member. Clicking the name of the cluster member navigates you to the page where you can change any of the configuration settings for that cluster member.

Runtime weight
Specifies the proportionate workload that is currently directed to the cluster member in comparison to the configured weight for that cluster member. To change the proportion, specify a new weight in the Runtime weight field and click the **Update** button for the Runtime weight column.

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 manage the members of a cluster. A cluster of application servers are managed together and participate in workload management.

A copy of the first cluster member that you create is stored as part of the cluster data and becomes the template for all additional cluster members that you create.

Any individual configuration change that you make to a cluster member does not affect the configuration settings of the cluster member template. You must use wsadmin commands to modify this template. Similarly, any changes that you make to the template does not affect existing cluster members. See the *Using the administrative clients* PDF for more information on how to modify this template.

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

Member name:

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

Data type	String

Weight:

Specifies the amount of work that is 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

Run in development mode:

Enabling this option may reduce the startup time of an application server. This may include JVM settings such as disabling bytecode verification and reducing JIT compilation costs. Do not enable this setting on production servers. This setting is only available on application servers running WebSphere Application Server Version 6.0 and later.

This option is not supported in an i5/OS environment.

Specifies that you want to use the JVM settings **-Xverify** and **-Xquickstart** on startup. After selecting this option, save the configuration and restart the server to activate development mode.

The default setting for this option is `false`, which indicates that the server will not be started in development mode. Setting this option to `true` specifies that the server will be started in development mode (with settings that will speed server startup time).

Data type Boolean
Default false

Parallel start:

Specifies whether to start the server on multiple threads. Starting the server on multiple threads might shorten the startup time.

Specifies that you want the server components, services, and applications to start in parallel rather than sequentially.

The default setting for this option is `true`, which indicates that the server be started using multiple threads. Setting this option to `false` specifies that the server will not be started in using multiple threads (which may lengthen startup time).

The order in which applications start depends on the weight you assign to each application. The application with the lowest starting weight starts first. Applications with the same starting weight start in parallel. You use the *Starting weight* field on the **Applications > Enterprise Applications >** *application_name* page of the administrative console to set the starting weight for an application.

Data type Boolean
Default true

Class loader policy:

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

Class loading mode:

Specifies whether the class loader should search in the parent class loader or in the application class loader first to load a class. The standard for Developer Kit class loaders and WebSphere Application Server class loaders is `Parent first`.

This field only applies if you set the Class loader policy field to `single`.

If you select `Parent last`, your application can override classes contained in the parent class loader, but this action can potentially result in ClassCastException or linkage errors if you have mixed use of overridden classes and non-overridden classes.

Process Id:

Specifies the native operating system process ID for this server.

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

Cell name:

Specifies the name of the cell in which this server is running.

The Cell name property is read only.

Node name:

Specifies the name of the node in which this server is running.

The Node name property is read only.

State:

Specifies the run-time execution state for this server.

The State property is read only.

Creating backup clusters

Use this task to configure a backup cluster that handles Enterprise JavaBeans (EJB) requests if the primary cluster fails.

Before you begin, create two clusters that are able to provide backup for each other. The objects and resources available in the primary cluster must also be available in the backup cluster. You must use the same cluster name, install the same applications, use the same application names, and define the same resources in the backup cluster as in the primary cluster.

The primary cluster and the backup cluster must reside in separate cells because a cluster must have a unique name within a cell. See “Backup clusters” on page 323 for more information.

Perform this task to create a backup cluster for your EJB clusters. When all the servers in the primary cluster fail, work is not halted because the backup cluster can continue serving requests for EJB work.

To configure a backup cluster, specify a name and a port. The port is called a domain bootstrap address and consists of a bootstrap host and port. The bootstrap host is the host that contains the deployment manager in which the backup cluster is configured. The bootstrap port is equal to the bootstrap port for the same deployment manager.

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

1. Determine the bootstrap host and port of the backup cluster.
   a. Connect the administrative console for the deployment manager that contains the backup cluster.
   b. Click **System administration > Deployment manager > Ports > BOOTSTRAP_ADDRESS**. The host and port for the BOOTSTRAP_ADDRESS instance is the host and port that the backup cluster uses. Remember these values for when you configure the primary cluster.
2. Connect the administrative console to the deployment manager that contains the primary cluster. Click **Servers > Clusters >** *cluster_name* **> Backup cluster**.
3. Ensure that the name of the backup cluster is the same as the primary cluster.
4. Click **Domain bootstrap address**. Specify the backup cluster deployment manager bootstrap host and port in the **Host** and **Port** fields. Click **OK**. The bootstrap host and port combined define a bootstrap address for the deployment manager. On the Domain Bootstrap Address page, use the **Configuration** tab to statically define the backup cluster; the static value is consumed each time the deployment manager starts. You can use the **Runtime** tab to define the backup cluster during run time only; when the deployment manager stops, the run-time backup cluster information is discarded.
5. Click **OK**.
6. Configure a core group bridge between each of the cluster core groups. Use an access point group to join the two core groups. In the deployment manager for the primary cell, configure an access point group that has a peer access point that refers to the core group access point in the backup cell. In the deployment manager for the backup cell, create an access point group that has the same name as the access point group that you created in the primary cell. Add a peer access point that refers to the core group access point in the primary cell. See “Configuring the core group bridge service” on page 404 for more information.

   Tip: If you are configuring a V5.x cluster to back up a cluster that is on the current release, do not configure the core group bridge service. Core groups are not supported in V5.x. Therefore, the V5.x cluster does not belong to a core group. The backup cluster still functions using only the domain bootstrap address.
7. Save the configuration.

The backup cluster completes EJB requests when the primary cluster fails.

If you experience problems when configuring your backup cluster, see the *Troubleshooting and support* PDF.

Backup clusters

Backup clusters mirror the primary server clusters. Mirrored cluster support is for Enterprise JavaBeans (EJB) requests only.

Overview and prerequisites

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

The fail back is automatic. You do not have to initiate fail back to the primary cluster after restarting the servers in the primary cluster. The backup cluster stops servicing requests as soon as the primary cluster becomes available. However, all of the deployment managers must be functional for backup cluster support, and the primary cluster must be defined as the backup for the backup cluster.

For the backup cluster to take over servicing requests successfully:

- The objects and resources available in the primary cluster must also be available in the backup cluster.
- You must use the same cluster name, install the same applications, use the same application names, and define the same resources in the backup cluster as in the primary cluster.
- The primary cluster and the backup cluster must reside in separate cells because a cluster must have a unique name within a cell.
- Both the primary and backup clusters must have a backup cluster configured, and each cluster must specify the opposite cluster as its backup cluster.

Because the primary and backup clusters reside in different cells, with the current version of WebSphere Application Server, the clusters also reside in different core groups. You must configure the core group bridge service to allow communication between core groups. The core group bridge service eliminates the requirement of a running deployment manager and a node agent for the backup cluster support. In the previous release, if the deployment manager stopped, new requests could not be forwarded to the backup cluster after the primary cluster failed. Any core group bridge server that is configured in the cell that contains the primary cluster can provide information about the backup cluster. The backup cluster support fails only if all of the core group bridge servers in a cell are not running.

For cluster failover and fail back to function as expected, all of the servers (deployment manager, node agents and application servers) for both the primary cluster and the backup cluster must be at a release and level that provides mirrored cluster support. However, if you are using a V5.x cluster to back up a cluster that is on the current release, you do not have the core group bridge service functionality available to you on the V5.x cluster.

Configuration

Mirrored cluster support is not configured by default. To use the mirrored cluster support, you must specify backup clusters in your configuration. Each cluster can have only one backup cluster, which must be configured before it is specified as a backup cluster.

To configure a backup cluster in a cluster, you must specify a name and a domain bootstrap address. The bootstrap host is the host that contains the deployment manager in which the backup cluster is configured. The bootstrap port is the bootstrap port for this deployment manager.

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

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

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

Because the primary and backup clusters reside in different cells, with the current version of WebSphere Application Server, the clusters also reside in different core groups. You must configure the core group bridge service to allow communication between core groups. Use an access point group to join the two core groups. In the deployment manager for the primary cell, configure an access point group that has the core group access point for the backup cell as a peer access point. In the deployment manager for the backup cell, create an access point group that has the same name as the access point group you created in the primary cell. Add a peer access point that refers to the core group access point in the primary cell.

See “Creating advanced core group bridge configurations” on page 414 for more information. If you are configuring a V5.x cluster to back up a cluster that is on the current release, you do not have to configure the core group bridge service because the V5.x cluster does not belong to a core group. The backup cluster still functions using only the domain bootstrap address.

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

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

In this sample, *ac* is the AdminClient, and *extendedCluster* is the ExtendedClusterObjectName for the primary cluster.

Fail back support scenarios

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

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

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

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

Backup cluster settings

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

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

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

Backup cluster name

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

Data type String

Domain bootstrap address settings

Use this page to specify the bootstrap address host and port of the deployment manager that contains the backup cluster. You must specify a value for the Host and Port fields and configure a core group bridge before you can use the backup cluster function.

When you start the deployment manager, the operating system uses the host and port values specified on the **Configuration** tab for the bootstrap address. If you need to change the host or port while the deployment manager is running, specify new values on the **Runtime** tab. Any values specified on this tab take affect as soon as they are saved. However, if you stop the deployment manager and then restart it, the values on the **Runtime** tab revert to the values that are specified on the **Configuration** tab when the deployment manager restarts.

To view this administrative console page, click **Servers** > **Clusters** > *cluster_name* > **Backup cluster** > **Domain bootstrap address**.

Host:

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

For example, if the host name is `myhost`, the fully qualified DNS name can be `myhost.myco.com` and the IP address can be `155.123.88.201`.

Data type String

Port:

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

Data type Integer

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.

Make sure that the members of your cluster have the debug port properly set. If multiple servers on the same node have the same debug port set, the cluster could fail to start. See “Java virtual machine settings” on page 285 for more information on how to change the debug port.

When you request that all members of a cluster start, the cluster state changes to *partially started* 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 *running*.

1. In the administrative console, click **Servers** > **Clusters**.
2. Select the 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 *running*. If the call to a node agent for a server fails, the server does not start.
   - **RippleStart** combines stopping and starting operations. It first stops and then restarts each member of the cluster. For example, your cluster contains 3 cluster members named *server_1*, *server_2* and *server_3*. When you click RippleStart, *server*_1 stops and restarts, then *server_2* stops and restarts,

and finally *server_3* stops and restarts. Use the RippleStart option instead of manually stopping and then starting all of the application servers in the cluster.

Important: If recently added cluster members do not start, you might not have selected Synchronize changes with Nodes when you added the members to the cluster. To determine if this is the problem:

a. In the administrative console click **Servers > Clusters**, select the cluster whose members did not start, and click **Stop**.
b. Click the name of the cluster and then click **Review**
c. Select Synchronize changes with Nodes, and then click **Save**.
d. Start the cluster and verify that all of the cluster members now start.

When you start the members of a cluster, you automatically enable workload management.

See Chapter 8, “Balancing workloads with clusters,” on page 305 for more information about the tasks that you can complete with clusters.

Stopping clusters

Use this task to stop a cluster and any application servers that are members of that cluster.

You can stop all application servers that are members of the same cluster at the same time by stopping the cluster.

1. Click **Servers > Clusters** in the console navigation tree to access the Server Cluster page.
2. Select 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 *partially stopped*. After all servers stop, the cluster state becomes Stopped.
   - **Immediate Stop** brings down the server quickly without regard to existing requests. The server ignores any current or pending tasks. When the stop operation begins, the cluster state changes to Partially stopped. After all servers stop, the cluster state becomes Stopped.

See Chapter 8, “Balancing workloads with clusters,” on page 305 for more information about the tasks you can complete with clustering.

Replicating data across application servers in a cluster

Use this task to configure a data replication domain to transfer data, objects, or events for session manager, dynamic cache, or stateful session beans. Data replication domains use the data replication service (DRS), which is an internal WebSphere Application Server component that performs replication services, including replicating data, objects, and events among application servers.

If you configured a data replication domain with a previous version of WebSphere Application Server, you might be using a multi-broker replication domain. Any replication domains that you create with the current version of WebSphere Application Server are data replication domains. Migrate any multi-broker replication domains to data replication domains. To learn the differences between the two types of replication domains, see “Comparison of multi-broker versus data replication domains” on page 333 and “Migrating servers from multi-broker replication domains to data replication domains” on page 331.

Use this task to configure *replication*, a service that transfers data, objects, or events among the application servers in a cluster. Use replication to prevent loss of session data with session manager, to further improve the performance of the dynamic cache service, and to provide failover in stateful session beans. For more information about replication, see “Replication” on page 329.

Important: If you select the **Configure HTTP memory-to-memory replication** option when you create a cluster, the replication domain is automatically created for you.

1. Create a replication domain. Use one of the following methods to create a replication domain:
   - **Create a replication domain manually.**

     To create a replication domain manually without creating a new cluster, click **Environment > Replication domains > New** in the administrative console.

     On this page you can specify the properties for the replication domain, including timeout, encryption, and number of replicas. See “Data replication domain settings” on page 330 for more information about the properties that you can configure for your replication domain.
   - **Create a replication domain when you create a cluster.**

     To create a replication domain when you create a cluster, click **Servers > Clusters > New** in the administrative console. Then click **Configure HTTP memory-to-memory replication**. The replication domain that is created has the same name as the cluster and has the default settings for a replication domain. The default settings for a replication domain are to create a single replica of each piece of data and to have encryption disabled. To modify the replication domain properties, click **Environment > Replication domains >** *replication_domain_name* in the administrative console. See “Creating clusters” on page 309 for more information about creating a cluster.

   For more information about the replication domain settings that you can configure in the administrative console, see “Data replication domain settings” on page 330
2. Configure the consumers, or the components that use the replication domains. Dynamic cache, session manager, and stateful session beans are the three types of replication domain consumers. Each type of consumer must be configured with a different replication domain. For example, session manager uses one domain and dynamic cache uses a different replication domain. However, use one replication domain if you are configuring HTTP session memory-to-memory replication and stateful session bean replication. Using one replication domain in this case ensures that the backup state information of HTTP sessions and stateful session beans are on the same application servers.
   - To configure dynamic cache replication, see the *Administering applications and their environment* PDF.
   - To configure memory-to-memory replication for session manager, see the *Administering applications and their environment* PDF.
   - To configure replication of stateful session beans, see the *Administering applications and their environment* PDF.
3. Determine whether your configuration requires additional thread resources.

   The replication service uses threads obtained from the Default thread pool for various tasks, including processing messages. Other application server components also use this thread pool. Therefore, during application server startup the default maximum thread pool size of 20 might not be sufficient to allow the replication service to obtain enough threads from the pool to process all of the incoming replication messages. The number of incoming messages is influenced by the number of application servers in the domain and the number of replication domain consumers on each application server. The number of messages to be processed increases as the number of application servers in the domain increases and/or the number of replication consumers increases.

   Persistent data not being replicated to the application servers during server startup might be an indication that you need to increase the setting for the maximum thread pool size. In larger configurations, doubling the maximum size of the Default thread pool to 40 is usually sufficient. However, if the number of application servers in a replication domain is greater ten and the number of replication domain consumers in each application server is greater than two, it might have to set the maximum thread pool size to a value greater than 40. See Thread pool settings for a description of how to change the maximum thread pool size setting.

Data is replicating among the application servers in a configured replication domain.

If you select DES or 3DES as the encryption type for a replication domain, an encryption key is used for the encryption of messages. At regular intervals, for example once a month, you should go to the **Environment > Replication domains** page in the administrative console, and click **Regenerate encryption key** to regenerate the key. After the key is regenerated, you must restart all of the application servers that are configured as part of the replication domain. Periodically regenerating the key improves data security.

Replication

Replication is a service that transfers data, objects, or events among application servers. Data replication service (DRS) is the internal WebSphere Application Server component that replicates data.

Use data replication to make data for session manager, dynamic cache, and stateful session beans available across many application servers in a cluster. The benefits of using replication vary depending on the component that you configure to use replication.

- Session manager uses the data replication service when configured to do memory-to-memory replication. When memory-to-memory replication is configured, session manager maintains data about sessions across multiple application servers, preventing the loss of session data if a single application server fails. For more information about memory-to-memory replication, see the *Administering applications and their environment* PDF.
- Dynamic cache uses the data replication service to further improve performance by copying cache information across application servers in the cluster, preventing the need to repeatedly perform the same tasks and queries in different application servers. For more information about replication in the dynamic cache, see the *Administering applications and their environment* PDF.
- Stateful session beans use the replication service so that applications using stateful session beans are not limited by unexpected server failures. For more information about stateful session bean failover, see the *Developing and deploying applications* PDF.

You can define the number of *replicas* that DRS creates on remote application servers. A replica is a copy of the data that copies from one application server to another. The number of replicas that you configure affects the performance of your configuration. Smaller numbers of replicas result in better performance because the data does not have to copy many times. However, if you create more replicas, you have more redundancy in your system. By configuring more replicas, your system becomes more tolerant to possible failures of application servers in the system because the data is backed up in several locations.

By having a single replica configuration defined, you can avoid a single point of failure in the system. However, if your system must be tolerant to more failure, introduce extra redundancy in the system. Increase the number of replicas that you create for any HTTP session that is replicated with DRS. Any replication domain that is used by dynamic cache must use a full group replica.

Session manager, dynamic cache, and stateful session beans are the three *consumers* of replication. A consumer is a component that uses the replication service. When you configure replication, the same types of consumers belong to the same *replication domain*. For example, if you are configuring both session manager and dynamic cache to use DRS to replicate objects, create separate replication domains for each consumer. Create one replication domain for all the session managers on all the application servers and one replication domain for the dynamic cache on all the application servers. The only exception to this rule is to create one replication domain if you are configuring replication for HTTP sessions and stateful session beans. Configuring one replication domain in this case ensures that the backup state information is located on the same backup application servers.

See the *Administering applications and their environment* PDF for more information on how to configure replication.

Replication domain collection

Use this page to view the configured replication domains that are used for replication by the HTTP session manager, dynamic cache service, and stateful session bean failover components. All components that need to share information must be in the same replication domain. Data replication domains replace multi-broker replication domains that were available for replication in prior releases. Migrated application servers use multi-broker replication domains which are collections of replicators. You should migrate any multi-broker replication domains to be data replication domains.

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

Name

Specifies a name for the replication domain. The name of the replication domain must be unique within the cell.

Domain type

Following are the two types of replication domains:

Multi-broker domain	Specifies a replication domain that was created with a previous version of WebSphere Application Server. This type of replication domain consists of replicator entries. Support of this type of domain remains for backward compatibility, but is deprecated. Multi-broker and data replication domains do not communicate with each other, so migrate any multi-broker replication domains to the new data replication domains. You cannot create a multi-broker domain or replicator entries in the administrative console after the deployment manager is upgraded to the current version of WebSphere Application Server.
Data replication domain	Specifies a replication domain created with the latest version of WebSphere Application Server. If the deployment manager has been upgraded to the latest version of WebSphere Application Server, you can create data replication domains only. With the data replication domain, you can specify a number of replicas instead of statically partitioning your replication settings. Specify a data replication domain for each consumer of the domain, for example, two separate domains for dynamic cache and session manager.

Data replication domain settings

Use this page to configure a data replication domain. Use data replication domains to transfer data, objects, or events for session manager, dynamic cache, or stateful session beans among the application servers in a cluster.

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

Name:

Specifies a name for the replication domain. The name must be unique within the cell.

Request timeout:

Specifies how long a replication domain consumer waits when requesting information from another replication domain consumer before it gives up and assumes the information does not exist.

Units	seconds
Default	5 seconds

Encryption type:

Specifies the type of encryption to use when transferring replicated data to another area of the network. Select NONE if you don't want to use encryption, DES if you want to use data encryption standard, or 3DES if you want to use triple DES. The default is NONE. The DES and 3DES options encrypt data sent between application server processes (for example, session manager and dynamic caching). Encrypting data improves the security of the network that joins the processes.

If you select DES or 3DES, after you click **Apply** or **OK**, a key for global data replication is generated. At regular intervals, for example once each month, you should navigate to this page in the administrative console and click **Regenerate encryption key** to regenerate this key. Periodically regenerating the key enhances security.

Data type	String
Default	NONE

Number of replicas:

Specifies the number of replicas that are created for every entry or piece of data that is replicated in the replication domain.

Single replica	One replica is created. This is the default value.
Full group replica	Each object is replicated to every application server that is configured as a consumer of the replication domain.
Specific number of replicas	A custom number of replicas for any entry that is created in the replication domain.

Migrating servers from multi-broker replication domains to data replication domains

Use this task to migrate multi-broker replication domains to data replication domains. Any multi-broker domains that exist in your WebSphere Application Server environment were created with a previous version of WebSphere Application Server.

For HTTP session affinity to continue working correctly when migrating V5.x application servers to V6.1 application servers, you must upgrade all of the Web server plug-ins for WebSphere Application Server to the latest version before upgrading the application servers that perform replication.

After you upgrade your deployment manager to the latest version of WebSphere Application Server, you can create data replication domains only. Any multi-broker domains that you created with a previous release of WebSphere Application Server are still functional, however, you cannot create new multi-broker domains or replicators with the administrative console.

The different versions of application servers cannot communicate with each other. When migrating your servers to the current version of WebSphere Application Server, keep at least two application servers running on the previous version so that replication remains functional.

Perform this task on any multi-broker domains in your configuration after all of your servers that are using this multi-broker domain have been migrated to the current version of WebSphere Application Server. For more information about the differences between multi-broker domains and the data replication domains, see "Comparison of multi-broker versus data replication domains" on page 333.

The following examples illustrate the migration process for common configurations:

Migrating an application server configuration that uses an instance of data replication service in peer-to-peer mode

Use this migration path to migrate a replication domain that uses the default peer-to-peer configuration. Dynamic cache replication domains use the peer-to-peer topology.

Before you begin, migrate all the Web server plug-ins for your application server cluster to the current version.

1. Migrate one or more of your existing servers to the current version of WebSphere Application Server.
2. In the administrative console, create an empty data replication domain. Click **Environment > Replication domains > New** in the administrative console.
3. Add your migrated application servers to the new data replication domain. For example, if you are migrating 4 servers, migrate 2 servers first and add them to the new replication domain. Configure the servers to use the new domain by configuring the consumers of the replication domain.
4. When the new data replication domains are successfully sharing data, migrate the rest of the servers that are using the multi-broker replication domain to data replication domains.
5. Delete the empty multi-broker replication domain.

Migrating an application server configuration that uses an instance of the data replication service in client/server mode

Use this set of steps to migrate a replication domain that uses client/server mode.

Before you begin migrating a client/server mode replication domain, consider if migrating your replication domains might cause a single point of failure. Because you migrate the servers to the new type of replication domain one at a time, you risk a single point of failure if there are 3 or fewer application servers. Before migrating, configure at least 4 servers that use multi-broker replication domains. Perform the following steps to migrate the multi-broker domains to data replication domains:

1. Migrate one or more of your existing servers to the current version of WebSphere Application Server.
2. In the administrative console, click **Environment > Replication domains > New** to create an empty data replication domain.
3. Add your migrated servers to the new data replication domain. For example, if you are migrating 4 servers, migrate two of these servers and then add them to the new replication domain. Configure the servers to use the new domain by configuring the consumers of the replication domain.
4. Add a part of the clients to the new data replication domain.
5. When the new data replication domains are successfully sharing data, migrate the rest of the clients and servers that are using the multi-broker replication domain to data replication domains.
6. Delete the empty multi-broker replication domain.

Migrating a replication domain that uses HTTP session memory-to-memory replication that is overloaded at the application or web module level

1. Upgrade your deployment manager to the current version of WebSphere Application Server. All the application servers remain configured with the old multi-broker domains on the previous version of WebSphere Application Server.
2. In the administrative console, create an empty data replication domain. Click **Environment > Replication domains > New** in the administrative console.
3. Migrate each application server to the current version of WebSphere Application Server, one at a time. The remaining servers on the previous version of WebSphere Application Server can still communicate with each other, but not with the migrated servers. The migrated servers can also communicate with each other.

4. Continue migrating all of the servers to the current version of WebSphere Application Server. All of the application servers are still using multi-broker replication domains, so the features of data replication domains cannot be used.
5. Configure all of the application servers to use the new data replication domain, adding the application servers to the empty replication domain that you created.
6. Restart all of the application servers in the cluster.
7. Delete the empty multi-broker replication domain.

During this process, you might lose existing sessions. However, the application remains active through the entire process, so users do not experience down time during the migration. Create a new replication domain for each type of consumer. For example, create one replication domain for session manager and another replication domain for dynamic cache.

Comparison of multi-broker versus data replication domains

Data replication domains replace multi-broker domains for data replication between application servers in a cluster.

transition:
Any replication domains that are created with a previous version of WebSphere Application Server might be multi-broker domains. Migrate any multi-broker domains to the new data replication domains. Although you can configure existing multi-broker domains with the current version of WebSphere Application Server, after you upgrade your deployment manager, you can create only data replication domains in the administrative console.

Multi-broker and data replication domains both perform the same function, which is to replicate data across the consumers in a replication domain. Configure all the instances of replication that need to communicate in the same replication domain. You can also configure the session manager with both types of replication domains to use topologies such as peer-to-peer and client/server to isolate the function of creating and storing replicas on separate application servers. You can control the redundancy of replication for each type of replication domain. With a data replication domain, you can specify a specific number of replicas.

If you used multi-broker domains with earlier releases of WebSphere Application Server, use the following comparison chart to learn the differences between how V5.x and V6.x application servers use the two types of replication domains:

	V5.x application servers using replication domains	**V6.x application servers using replication domains**
Replication domain types	Uses only multi-broker replication domains for replication.	Servers that are using the current version of WebSphere Application Server can be configured to use both multi-broker replication domains and data replication domains for replication. The two types of domains provide backward compatibility with multi-broker domains that were created with a V5.x server. You should migrate any multi-broker domains to data replication domains.

	V5.x application servers using replication domains	V6.x application servers using replication domains
Data transport method	Uses multi-broker domain objects that contain configuration information for the internal Java Message Service (JMS) provider, which uses JMS brokers as replicators.	Uses data replication domain objects that contain configuration information to configure the high availability framework on WebSphere Application Server. The transport is no longer based on the JMS API. Therefore, no replicators and no JMS brokers exist. You do not have to perform the complex task of configuring local, remote, and alternate replicators. The earlier version of WebSphere Application Server did not support data replication domains. The current version of WebSphere Application Server can be configured to perform replication using old multi-broker domains by ignoring any JMS-specific configuration and by using the other parameters to configure replication through the high availability framework.
Replication domain configuration	The earlier version of WebSphere Application Server encourages the sharing of replication domains between different consumers, such as session manager and dynamic cache.	The current version of WebSphere Application Server encourages creating a separate replication domain for each consumer. For example, create one replication domain for session manager and another replication domain for dynamic cache. The only situation where you should configure one replication domain is when configuring session manager replication and stateful session bean failover. Using one replication domain in this case ensures that the backup state information of HTTP sessions and stateful session beans are on the same application servers.
Partial partitioning	You can configure partial partitioning. Partition the replication domain to filter the number of processes to send data.	Partial partitioning is deprecated. When using data replication domains, you can specify a specific number of replicas for each entry. However, if you specify a number of replicas larger than the number of backup application servers that are running, the number of replicas is the number of application servers that are running. After the number of application servers increases above your configured number of replicas, the number of replicas that are created is equal to the number that you specified.

	V5.x application servers using replication domains	V6.x application servers using replication domains
Domain sharing	Multiple data replication service (DRS) instances share multi-broker domains. A limitation exists on the number of multi-broker domains that you can create because every multi-broker domain contains at least one replicator. A maximum of one replicator can be on each application server.	All DRS instances in a replication domain use the same mode. Each replication domain must contain either client only and server only instances, or client and server instances only. For example, if one instance is configured to client and server, all other instances must be client and server. If one instance in a replication domain is configured to be a client only, you can add client only and server only instances, but not a client and server instance.

To migrate multi-broker domains to data replication domains, see the *Migrating, coexisting, and interoperating* PDF.

Replicating data with a multi-broker replication domain

Use this task to mange replication domains that you migrated from a WebSphere Application Server v5 environment.

Multi-broker replication domains are not created in WebSphere Application Server v6.x environments. However they can be migrated from existing WebSphere Application Server v5.x environments. If you migrate v5.x multi-broker replication domains, you can use the Multi-broker domain panel in the v6.x administrative console to managed these domains.

Although you can manage migrated multi-broker domains with the current version of WebSphere Application Server, after you upgrade your deployment manager, you can create only data replication domains in the administrative console. Consider migrating any existing multi-broker domains to the new data replication domains. See “Migrating servers from multi-broker replication domains to data replication domains” on page 331 and “Multi-broker replication domains” on page 336 for more information about the benefits of migrating your replication domains.

If you do not have any existing replication domains, see “Replicating data across application servers in a cluster” on page 327 for information about creating new data replication domains.

If you are performing this task, it is assumed that you configured replication with a previous version of WebSphere Application Server and defined replication domains that list connected replicator entries (residing in managed servers in the cell) that can exchange data. You can manage these existing replication domains and replicator entries, but you cannot create new multi-broker replication domains or new replicator entries in the administrative console.

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 relationship exists between replicators and application servers. During configuration, you can select the local replicator as the default replicator.

1. Manage multi-broker replication domain configuration settings. In the administrative console, click **Environment > Replication domains**.
2. Click on a **Multi-broker domain**, and update the values for a particular multi-broker replication domain. The default values are generally sufficient, especially for the 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 HTTP session if the process maintaining the HTTP session fails. Partitioning is not supported for sharing of cached data that is 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 HTTP session if the process maintaining the HTTP session 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 replication service.

3. Maintain the replicators that you have already defined. You cannot create any new replicators. 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. In the administrative console, click **Environment > Replication domains >** *replication_domain_name* **> Replicator entries >** *replicator_entry_name*.
   b. Specify a replicator name and select a server available within the cell to which you can assign a replicator. Also specify a host name and ports. Note that a replicator has two ports (replicator and client ports) 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 **Regenerate encryption key** on the settings for a replication domain instance at regular intervals, such as monthly.

   Periodically changing the key enhances security.

Multi-broker replication domains

A multi-broker replication domain is a collection of replication entries, or *replicator* instances, used by clusters or individual servers within a cell. Multi-broker replication domains were created with a previous release of WebSphere Application Server.

Note: After you upgrade your deployment manager to the latest version of WebSphere Application Server, you can create data replication domains only. Any multi-broker domains that you created with a previous release of WebSphere Application Server are still functional, however, you cannot create new multi-broker domains or replicator instances with the administrative console. See “Comparison of multi-broker versus data replication domains” on page 333 for more information.

A replication entry, or replicator, is a run-time component that handles the transfer of internal WebSphere Application Server data. All replicators within a replication domain connect with each other, forming a network of replicators.

Components such as session manager and dynamic cache can connect to any replicator within a domain to receive data from their peer components on other application servers that are connected other replicators in the same domain. If the replicator that a component is connected to fails, the component automatically attempts to reconnect to another replicator in the domain and recover any data that was missed while the component was not connected to a replicator.

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 a given replication domain across a cell. Most default settings tune and control the behavior of replicator entries that are 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. 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 HTTP session 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 data that is no longer valid and actual cached data maintained by 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.

Multi-broker replication domain settings

Use this page to configure a multi-broker replication domain. This administrative console page applies only to replication domains that were created with a previous version of WebSphere Application Server. Replication domains use the data replication service (DRS).

To view this administrative console page, click **Environment > Replication domains >** *multibroker_replication_domain_name*.

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

Name:

Specifies a name for the replication domain. The name must be unique within the cell.

Request timeout:

Specifies the number of seconds that a replication domain consumer waits when requesting information from another replication domain consumer 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 the object transfers over the network. The options include NONE, DES, TRIPLE_DES. The default is NONE. The DES and TRIPLE_DES options encrypt data sent between WebSphere Application Server processes and 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 **Regenerate encryption key** at regular intervals such as monthly because periodically changing the key enhances security.

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. There should be at least one server listening to every partition. If there are no servers listening on a partition, all the replicas created in that partition are lost because there is no server to cache the objects. The default partition size is 10, and the partition size should be 10 or more to enhance performance.

Partitioning the replication domain is only applicable if you are replicating data to support retrieval of an HTTP session if the process maintaining the HTTP session 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 role or runtime mode for a server. This role or mode affects whether a WebSphere Application Server 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 is made. Use this option only if you are using session manager with memory to memory replication. Enable this option if you are replicating data to support retrieval of an HTTP session if the process maintaining the HTTP session fails. This option restricts the recipient of the data to a single instance.

Note: Do not enable this option on a domain that is using dynamic cache replication.
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.

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 Java 2, Enterprise Edition (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 instantiate the object on the receiving side so it 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 the class definitions do not need to be stored on the receiving side. Or, the option requires that you move class definitions from the Web application class path to the system class path.

DRS pool size:

Specifies the size of the pool of resources allocated for communication with its Java Message Service (JMS) transport. You must configure this number to be the same as the DRS partition size. The default is 10.

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

DRS pool connections:

Specifies that the domain replication service should create a pool of connections with its Java Message Service (JMS) transport rather than reusing a single connection. You can pool connections when using a single replica or client server environment. You should not pool connections in a peer to peer environment.

The default is to not create a pool of connections for replication.

Replicator entry collection:

Use this page to view and manage replicator entries. Replicator entries are for use only with multi-broker replication domains. Each multi-broker replication domain consists of one or more replicator entries.

To view this administrative console page, click **Environment > Replication domains >** *replication_domain_name* **> Replicator entries**.

Replicator entries are only valid for multi-broker domains, which are replication domains created with a previous version of WebSphere Application Server. When you migrate your deployment manager to the current version of WebSphere Application Server, you are no longer be able to create new replicator entries in the administrative console. You can only view and modify settings for replicator entries that were created with the previous version of WebSphere Application Server.

Replicator name:

Specifies a name for the replicator entry.

Replicator entry settings:

Use this page to view and configure a replicator entry (or *replicator*). Replicators are used with multi-broker replication domains.

To view this administrative console page, click **Environment > Replication domains >** *replication_domain_name* **> Replicator entries >** *replicator_entry_name*.

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

Replicator name:

Specifies a name for the replicator entry.

Server:

Specifies the server for which you are defining a replicator. You can view the names of servers that do not already have replicators. You can create a maximum of one replicator on any application server.

Replicator and client host name:

Specifies the IP address, domain name service (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JavaServer Pages (JSP) file, or HTML page).

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

Replicator Port:

Specifies the port for which the replicator is configured to accept messages from other replicators. 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.

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.

Deleting clusters

Use this task to remove a cluster and all of its cluster members.

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

Tip: If the cluster you are removing has applications or modules mapped to it, remap the modules to another cluster, or create a new cluster and remap the modules to the new cluster, before removing the old cluster. After a cluster to which modules are mapped is deleted, the modules cannot be remapped to another cluster. Therefore, if you do not remap the modules to another cluster before deleting the old one, you must uninstall all of the modules that were mapped to the old cluster, and then reinstall them on a different cluster.

1. In the administrative console, click **Servers > Clusters**.
2. Make sure the cluster you want to remove is **stopped**. If the cluster is **started**, stop the cluster..
3. Delete the cluster. Select the cluster you want to delete, and click **Delete**.
4. Click **Save** to save your configuration changes. As part of saving your change to the configuration, select Synchronize changes with Nodes before you click **Save**.

The cluster and all of the cluster members are deleted.

Deleting specific cluster members

Use this task to remove a cluster member from an existing cluster. Removing a cluster member deletes the associated application server.

You must delete an application server to remove it from a cluster.

Note: If, in the administrative console, you select **Include cluster members in the collection** as one of your console page preferences for the Applications server page, you can use either the Application servers page or the Cluster members page to delete an application server.

To use the Cluster members page to remove an application server from a cluster:

1. In the administrative console, click **Servers > Clusters**.
2. Click the name of the cluster that contains the cluster member that you are removing from the cluster, and then click **Cluster members**.
3. Check the status of the cluster member that you are removing. If the cluster member is started, select the cluster member, and click **Stop**, and then view the Status of this cluster member again, along with any messages or logs to make sure the cluster member stops. You cannot remove a cluster member while it is running.
4. Delete the cluster member. Select the cluster member you want to delete, and click **Delete**.
5. Click **Save** to save your configuration changes. As part of saving your change to the configuration, select Synchronize changes with Nodes before you click **Save** .

The cluster member is deleted.

Tuning a workload management configuration

You can set values for several workload management client properties to tune the behavior of the workload management runtime.

You set the properties as command-line arguments for the Java virtual machine (JVM) process in which the workload management client is running.

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

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

1. Access the Java Virtual Machine page.
   a. In the administrative console, click **Servers > Application Servers >** *server_name***> Java and Process Management > Process Definition**.
   b. On the Process Definition page, click **Java Virtual Machine**.
2. On the Java Virtual Machine page, specify one or more of the following command-line arguments in the Generic JVM arguments field:
   - **-Dcom.ibm.CORBA.RequestTimeout=*timeout_interval***

     If your application is experiencing problems with timeouts, this argument changes the value for the com.ibm.CORBA.RequestTimeout property, which specifies the timeout period for responding to requests sent from the client. This argument uses the -D option. *timeout_interval* is the timeout period in seconds. If your network experiences extreme latency, specify a large value to prevent timeouts. If you specify a value that is too small, an application server that participates in workload management can time out before it receives a response.

     Important: Be careful specifying this property; it has no recommended value. Set it only if your application is experiencing problems with timeouts.
   - **-Dcom.ibm.websphere.wlm.unusable.interval=*interval***

     If the workload management state of the client is refreshing too soon or too late, this argument changes the value for the com.ibm.websphere.wlm.unusable.interval property, which specifies the time interval that the workload management client runtime waits after it marks a server as unavailable before it attempts to contact the server again. This argument uses the -D option. *interval* is the time in seconds between attempts. The default value is 300 seconds. If the property is set to

a large value, the server is marked as unavailable for a long period of time. This prevents the workload management refresh protocol from refreshing the workload management state of the client until after the time period has ended.

3. Click **OK** and **Save** to save your configuration changes.
4. Stop the application server and then restart the application server.

Workload management runtime exceptions

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

The workload management service might create the following exceptions if it encounters problems:

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

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

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

Clustering and workload management: Resources for learning

Use the following links to find relevant supplemental information about clustering and workload management. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

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

View links to additional information about:

- Programming model and decisions
- Programming instructions and examples

Programming model and decisions

- IBM WebSphere V5.0 Performance, Scalability, and High Availability: WebSphere Handbook Series
- IBM WebSphere V5.0 Performance, Scalability, and High Availability: WebSphere Handbook Series at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/SG246198.html
- IBM WebSphere Application Server V5.0 System Management and Configuration: WebSphere Handbook Series at http://publib-b.boulder.ibm.com/redbooks.nsf/RedbookAbstracts/sg246195.html?Open

Programming instructions and examples

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

Chapter 9. Setting up a high availability environment

The application server infrastructure that is managed by a high availability manager includes cells and clusters. These components relate closely to core groups, high availability groups, and the policy that controls the high availability infrastructure.

Plan out how you need to set up your high availability environment to avoid the risk of a failure without failover coverage. As part of the planning process, understand how the high availability manager can assist you in controlling this type of an environment.

The high availability manager is designed to function in all of the supported WebSphere Application Server topologies. However, a high availability-managed environment must comply to the following rules:

- A cell in a high availability infrastructure is partitioned into one or more core groups. WebSphere Application Server provides a default core group as part of the high availability manager function. Additional core groups can be created using the administrative console.
- A core group cannot extend beyond the boundaries of a cell, and it cannot overlap with any other core groups.
- A cluster must be a member of only one core group. All of the individual members of that cluster must be members of the same core group.
- Individual application servers are also members of a core group.
- All running members of a core group must be able to communicate with all of the other running members of that same core group.

While administering your core groups, you might need to perform one or more of the following tasks. These tasks can be performed in any order.

1. Set up preferred coordinators.
2. Change the number of core group coordinators.
3. Configure a core group transport.
4. Configure the discovery protocol for a core group.
5. Configure the failure detection protocol for a core group.
6. Configure a core group for replication
7. Configure core group IP caching.
8. Set up IP addresses for high availability manager communications for V6.0.2 and higher.
9. Configure core group socket buffers.
10. Specify a core group when adding a node.
11. Specify a core group when creating an application server.
12. View the core groups contained in a cell.
13. View the members of a core group.
14. Create a new core group.
15. Move core group members to a different core group.
16. View high availability group information
17. Create a new policy and associate it with a high availability group.
18. Change the policy associated with a high availability group.
19. Route high availability group work to a different server.
20. Create core group access points if you create additional core groups.
21.

Important: After you set up your WebSphere Application Server environment to comply with all of the high availability-managed environment rules, use the default core group to control this environment. DO NOT add additional core groups unless your environment absolutely requires them. Also, do not change the default configurations unless you are doing so to solve a specific problem or situation. When you do make configuration changes, such as changing the policy for a high availability group or moving core group members between core groups in a multi-core group environment, make sure you fully understand the effect such changes will have on your entire environment.

Troubleshoot high availability environment problems. See the *Troubleshooting and support* PDF for more information.

High availability manager

WebSphere Application Server includes a high availability manager component. The services that the high availability manager provides are only available to WebSphere Application Server components.

A high availability manager provides several features that allow other WebSphere Application Server components to make themselves highly available. A high availability manager provides:

- A framework that allows singleton services to make themselves highly available. Examples of singleton services that use this framework include the transaction managers for cluster members, and the default IBM messaging provider, commonly referred to as a messaging engine.
- A mechanism that allows servers to easily exchange state data. This mechanism is commonly referred to as the bulletin board.
- A specialized framework for high speed and reliable messaging between processes. This framework is used by domain replication services when WebSphere Application Server is configured for memory-to-memory replication.

A high availability manager instance runs on every application server, proxy server, node agent and deployment manager in a cell. A cell can be divided into multiple high availability domains known as core groups. Each high availability manager instance establishes network connectivity with all other high availability manager instances in the same core group, using a specialized, dedicated, and configurable transport channel. The transport channel provides mechanisms which allow the high availability manager instance to detect when other members of the core group start, stop, or fail.

Within a core group, high availability manager instances are elected to coordinate high availability activities. An instance that is elected is known as a core group coordinator. The coordinator is highly available, such that if a process that is serving as a coordinator stops or fails, another instance is elected to assume the coordinator role, without loss of continuity.

Highly available components

A *highly available component* is a component for which a high availability group is defined on the processes where that component can run. The coordinator tracks high availability group membership, and knows on which processes each highly available component can run.

The coordinator also associates a high availability policy with each high availability group. A *high availability policy* is a set of directives that aid the coordinator in managing highly available components. For example, a directive might specify that a component runs on a specific process, if that process is available. Directives are configurable, which makes it possible for you to tailor policies to your installation.

The coordinator is notified as core group processes start, stop or fail and knows which processes are available at any given time. The coordinator uses this information, in conjunction with the high availability group and policy information, to ensure that the component keeps functioning. The coordinator uses the policy directives to determine on which process it starts and runs each component. If the chosen process

fails, the coordinator restarts the component on another eligible process. This reduces the recovery time, automates failover, and eliminates the need to start a replacement process.

State data exchange

The high availability manager provides a specialized messaging mechanism that enables processes to exchange information about their current state. Each process sends or posts information related to its current state, and can register to be notified when the state of the other processes changes. The Work Load Management (WLM) component uses this mechanism to build and maintain routing table information. Routing tables built and maintained using this mechanism are highly available.

Replication

WebSphere Application Server provides a domain replication service (DRS) that is used to replicate HTTP session data, stateful EJB sessions, and dynamic cache information among cluster members. When DRS is configured for memory-to-memory replication, the transport channels defined for the high availability managers are used to pass this data among the cluster members.

When to use a high availability manager

A high availability manager consumes valuable system resources, such as CPU cycles, heap memory, and sockets. These resources are consumed both by the high availability manager and by WebSphere Application Server components that use the services that the high availability manager provides. The amount of resources that both the high availability manager and these WebSphere Application Server components consume increases exponentially as the size of a core group increases. For large core groups, the amount of resources that the high availability manager consumes can become significant. If the services that the high availability manager provides are not used, then disabling the high availability manager frees these resources.

The capability to disable the high availability manager is most useful for large topologies where none of the high availability manager provided services are used. In certain topologies, only some of the processes use the services that the high availability manager provides. In these topologies, you can disable the high availability manager on a per-process basis, which optimizes the amount of resources that the high availability manager uses.

Do not disable the high availability manager on administrative processes, such as node agents and the deployment manager, unless the high availability manager is disabled on all application server processes in that core group.

Some of the services that the high availability manager provides are cluster based. Therefore, because cluster members must be homogeneous, if you disable the high availability manager on one member of a cluster, you must disable it on all of the other members of that cluster.

When determining if you must leave the high availability manager enabled on a given application server process, consider if the process requires any of the following high availability manager services:

- Memory-to-memory replication
- Singleton failover
- Workload management routing
- On-demand configuration routing

Memory-to-memory replication

Memory-to-memory replication is a cluster-based service that you configure or enable at the application server level. If memory-to-memory replication is enabled on any cluster member, then the high availability manager must be enabled on all of the members of that cluster. Memory-to-memory replication is automatically enabled if:

- Memory-to-memory replication is enabled for Web container HTTP sessions. See the *Developing and deploying applications* PDF for more information.
- Cache replication is enabled for the dynamic cache service. See the *Developing and deploying applications* PDF for more information.
- EJB stateful session bean failover is enabled for an application server.See the *Developing and deploying applications* PDF for more information.

Singleton failover

Singleton failover is a cluster-based service. The high availability manager must be enabled on all members of a cluster if:

- The cluster is configured to use the high availability manager to manage the recovery of transaction logs. See the *Troubleshooting and support* PDF for more information about the recovery of transaction logs.
- One or more instances of the default Java Message Service (JMS) provider are configured to run in the cluster. The default JMS provider is the messaging engine that is provided with WebSphere Application Server.

Workload management routing

Workload management (WLM) propagates the following classes or types of routing information:

- Routing information for enterprise bean Internet Inter-ORB Protocol (IIOP) traffic.
- Routing information for the default IBM Java Messaging Service (JMS) provider, which is also referred to as the messaging engine.

WLM uses the high availability manager to both propagate the routing information and make it highly available. Although WLM routing information normally applies to clustered resources, it can also apply to non-clustered resources, such as standalone messaging engines. Under normal circumstances, you must leave the high availability manager enabled on any application server that produces or consumes either IIOP or messaging engine routing information. For example if:

- The routing information producer is an enterprise bean application that resides in cluster 1.
- The routing information consumer is a servlet that resides in cluster 2.

When the servlet in cluster 2 calls the enterprise bean application in cluster 1, the high availability manager must be enabled on all servers in both clusters.

Workload management provides an option to statically build and export route tables to the file system. Use this option to eliminate the dependency on the high availability manager. See “Enabling static routing for a cluster” on page 317 for more information about the Export route table option.

On-demand configuration routing

In a Network Deployment system, the on-demand configuration is used for both proxy server and Web services routing. The high availability manager must be enabled on all processes to which the proxy server routes work, and on all processes that are running Web services.

Best practices

Although not required, core groups are usually homogeneous. If you have a large installation and you need to set up a mix of processes that do and do not use the high availability manager, you can:

- Create a new core group and move all application servers on which the high availability manager is disabled to this core group. A core group that contains only application servers on which the high availability manager is disabled does not have to contain an administrative process and have no restrictions on how large such a core group can become.

- Leave the remaining applications servers that require high availability manager services in the default core group. You can create additional core groups for some of the application servers that require the high availability manager if the default core group becomes too large.

Core groups

A core group is a high availability domain that consists of a set of processes in the same cell that can directly establish high availability relationships. Highly available components can only fail over to another process in the same core group and replication can occur only between members of the same core group.

A cell must contain at least one core group, although multiple core groups are supported. Each core group contains a core group coordinator to manage its high availability relationships, and a set of high availability policies that are used to manage the highly available components within that core group.

Core group members

Every deployment manager, node agent, application server, and proxy server is a member of a core group. When a process is created it is automatically added to a core group. The core group membership is stored in a WebSphere Application Server configuration document. You can move processes from one core group to another. The following rules govern the core group membership:

- Every process is a member of exactly one core group.
- All members of a cluster must be members of the same core group.
- Each core group must contain at least one node agent or the deployment manager.

A core group member has a well-defined life cycle. When the first core group member starts, the transport that is dedicated to that core group automatically starts. The Discovery Protocol, View Synchrony Protocol, and Failure Detection Protocol for that core group member also start and run for the entire lifetime of the core group member:

- The Discovery Protocol is responsible for discovering when other core group processes start, and for opening network connections to these other members.
- The View Synchrony Protocol is responsible for establishing reliable messaging with other core group members after the connections are opened.
- The Failure Detection Protocol is responsible for detecting when other core group members stop or become unreachable because of a network partition.

Core group coordinator

The core group coordinator is responsible for coordinating high availability activities between the core group members for which View Synchrony Protocol is established.

Core group transport

Network communication between all the members of a core group is essential. The network environment must consist of a fast local area network (LAN) with full Internet Protocol (IP) visibility and bidirectional communication between all core group members. Each core group member must be able to receive communications from any of the other core group members.

Multiple core groups

A cell, by default, contains a single core group, called DefaultCoreGroup. All processes in the cell are initially members of this core group. A single core group is usually sufficient. However, some topologies or special circumstances require multiple core groups. There are also topologies that do not require multiple core groups but having them is a good practice. For example, you might want to define multiple core groups if :

- There are one or more firewalls within a cell. A core group can not contain members from multiple firewall protection domains.
- A large number of processes in the cell and the core group protocols, such as the View Synchrony Protocol, consume correspondingly large amounts of resources such as CPU.
- Core group protocols, such as the Failure Detection Protocol, need tuning or configuring to use values that work best with smaller numbers of core group members.

If members of different core groups need to share WLM routing information, use the core group bridge service to connect these core groups. The core group bridge service uses access point groups to connect the core groups. A core group access point defines a set of bridge interfaces that resolve to IP addresses and ports. The core group bridge service uses this set of bridge interfaces to enable members of one core group to communicate with members of another core group.

Core group coordinator

Every core group has a coordinator that manages high availability activities between the core group members. The coordinator manages the failover of highly available singleton services and distributes live server state data to interested core group members. The coordinator uses some CPU and memory (JVM heap) resources to perform these tasks. In some configurations, the amount of resources that the coordinator uses might be large.

The coordinator workload can be divided over multiple coordinator instances. Each instance runs on a different core group member and is assigned a portion of the overall coordination workload. Dividing the workload across multiple coordinator instances enables you to share the associated resource costs across machines. The coordinator function remains highly available, regardless of how its workload is divided or assigned to core group members.

Coordinator election

When a core group member starts or stops, the View Synchrony Protocol installs a new view. The view consists of the core group members that are connected and cooperating. Whenever a new view is installed, it might be necessary to redivide the coordinator workload among the core group members. For example, a core group member that is hosting a coordinator instance might fail and the high availability manager must elect a replacement coordinator.

Informational messages, similar to the following message, are logged in the SystemOut.log file when a particular core group member is elected as a coordinator:

```
HMGR0206I: The coordinator is an active coordinator for core group DefaultCoreGroup
```

Messages, similar to the following message, are logged if a core group member is no longer an elected coordinator:

```
HMGR0207I: The coordinator was previously an active coordinator for core group
           DefaultCoreGroup but has lost leadership.
```

Important: Remember that coordinator election occurs whenever the view changes. Electing a new coordinator uses a lot of resources because this process causes increased network traffic and CPU consumption. Specifying a preferred coordinator server, whenever practical, helps eliminate the need to make frequent coordinator changes.

Multiple coordinators

The WebSphere core group configuration data contains a field in which users can specify the number of coordinators. The default value for this field is 1. This default value is sufficient for most installations and applications. Use multiple coordinators when the core group member that is selected as the coordinator uses noticeably more memory or CPU than similar core group members. In addition, some software

products that heavily use the high availability framework instruct you to increase the number of coordinators.

Preferred servers

When you configure a core group, you can specify the core group members the high availability manager should use as coordinators, if they are available. Preferred coordinator servers should be core group processes that are cycled as infrequently as possible. The preferred coordinator servers should also be hosted on machines with excess capacity.

Specifying preferred coordinator servers is a good practice. When coordinators are elected during a view change, the high availability manager checks for a list of preferred servers. If there is a list, the high availability manager selects a server from that list as the coordinator. If there is no list, the high availability manager selects the view member with the lexically lowest name as the coordinator, which incurs some overhead if it causes the coordinator to move.

Core group administration considerations

Core group configuration information is stored in a CoreGroup configuration object that is backed by a `coregroup.xml` document. Process-specific configuration information for each core group member is stored in a HAManagerService configuration object that is backed by a `hamanagerservice.xml` document.

The `coregroup.xml` document is a cell-scoped document. The master copy of this document is stored in the configuration repository for the deployment manager. A copy of this document is shadowed to every node in the cell. The `coregroup.xml` document includes of the following configuration information:

- The list of core group members
- The high availability policies for the core group
- The core group coordinator configuration information
- The core group transport configuration information

The core group member process-specific configuration information stored in the `hamanagerservice.xml` document includes:

- Whether the high availability manager is enabled.
- The transport buffer size.
- The name of the core group to which the member belongs.
- How frequently the high availability manager checks the health of highly available singletons running on the member, if a length of time is in affect for this function.

Core group configuration document

The master copy of the core group configuration document is directly modified when direct attributes, such as the coordinator configuration, are modified. The master copy of the core group configuration document is implicitly modified when a server is created or deleted, or a node is added or removed. In either case, the list of core group members is updated to reflect which processes are added or removed.

The set of core group members for which the View Synchrony Protocol is established is commonly referred to as a *view*. Whenever a view is installed, one of the core group members is elected to send its current configuration to all other members of the view. This processing ensures that all members of the view are running with a consistent core group configuration. This processing also means that inconsistencies in a high availability policy or coordinator configuration are tolerated. However, inconsistencies in the list of core group members or the core group transport are not tolerated.

Before you modify a list of core group members, remember that all core groups must contain at least one administrative process. In a situation where the configuration document is synchronized to all of the nodes in the cell, and you have multiple core group processes running, the running core group administrative processes are notified whenever the configuration document is modified. The high availability manager

selects one of the administrative processes to reread the configuration and distribute the updated configuration to all of the other core group members in the same view. These changes are then dynamically picked up by all of these other core group members. If the core group does not contain at least one administrative process that is running when a configuration change is made, the updated configuration is not properly passed on to the core group members.

If you modify a list of core group members, do not start a member of that core group until you are sure that the change is fully synchronized to all nodes in the cell. If a node agent is down when the configuration change is made, you must manually synchronize the configuration change before any processes are started on that node. If you do not manually synchronize the change, the process that is starting cannot establish the View Synchrony Protocol with the other core group members because when a core group member starts, it reads the core group configuration information from the repository on the local node. It then opens connections to other core group members and attempts to establish the View Synchrony Protocol with them. If the local copy of the `coregroup.xml` document is not synchronized with the master core group configuration document, problems occur. For example, if the running processes dynamically reloaded the updated configuration, the configuration for the process that just started is out of sync with the configurations of the other core group members. If the update changed the list of core group members, the list is now inconsistent across the nodes in the cell, and any attempt to establish view synchrony fails because of these inconsistent member lists. When this condition is detected, an error message similar to the following message is logged:

```
DCSV8022I: DCS Stack {0} at Member {1}: Inconsistency of configured defined set
with that of another member. Inconsistent member is {2}. The list of members only
in the local defined set is {3}, whereas the list of members only in the defined
set at the inconsistent member is {4}.
```

When a process detects an inconsistent core group membership condition, the process attempts to reread the core group configuration several times. It is possible that the configuration document is in the process of being synchronized to the node. In such a case, rereading the configuration document can resolve the inconsistency. However, if the process can not resolve the inconsistency after trying to reread the configuration several times, the process stops trying to resolve the inconsistency. To recover from this situation, you must resynchronize the configuration and restart the process.

Core group process-specific configuration document

Unlike the cell-scoped core group configuration information that is contained in the `coregroup.xml` document, the process-specific configuration information for each core group member that is contained in the `hamanagerservice.xml` document cannot be dynamically reloaded. You must restart a process before core group process-specific configuration changes go into affect.

Core group scaling considerations

The amount of system resources, such as CPU and memory, that the high availability manager consumes does not increase linearly as the size of a core group increases. For example, the View Synchrony Protocol that the high availability manager uses requires a large amount of these resources to maintain a tight coupling over the core group members. Therefore, the amount of resources that a large core group consumes might become significant.

View Synchrony Protocol resource requirements can vary considerably for different core groups of the same size. The amount of resources that the View Synchrony Protocol uses for a core group is determined by:

- The number of applications that are running.
- The type of applications that are running.
- The high availability manager services that are used.

When setting up core group scalability, you must ensure that:

- All of the processes within the cell are distributed properly into core groups of appropriate sizes. Properly distributing these processes limits the amount of resources that the View Synchrony Protocol consumes.
- All of the processes within a given core group are properly configured to support the high availability services that are used within the core group.

Consider implementing one or more of the following scalability techniques to scale the high availability manager in large cells, even if your system is operating properly. The two most basic techniques are:

- Disabling the high availability manager if it is not required.
- Distributing the processes over a number of core groups and using a core group bridge to connect the core groups as required.

Adjusting the size of a core group

Core group size directly effects three aspects of high availability manager processing that impact resource usage:

- The first and most significant aspect is the establishment of the View Synchrony Protocol over a set of active core group members. This activity is commonly referred to as a *view change*.
- The second aspect is the regularly scheduled discovery and failure detection tasks that high availability manager runs in the background.
- The third aspect is the resource usage that results when other WebSphere Application Server components use high availability manager-provided services.

View Changes

The View Synchrony Protocol creates a new view whenever it detects that there is a change in core group members that are active. A view change typically occurs whenever a core group member starts or stops. When a core group member starts, it opens a connection to all of the other running core group members. When a core group member stops, other core group members detect that their open connections to the stopped member are closed. In either case, the View Synchrony Protocol needs to account for this change. In the case of a newly started member, the View Synchrony Protocol must establish a view that includes the new member. In the case of a stopped member, the View Synchrony Protocol must establish new view for the surviving core group members that excludes the stopped member.

Establishing a new view is an important activity but uses a lot of system resources, especially for large core groups.

- Each running core group member must communicate its current state to other core group members, including information about the messages it has sent or received in the current view.
- All messages sent in a given view must be received and acknowledged by all recipients before a new view can be installed. Under normal operating conditions, receipt of these messages is acknowledged slowly. Completing messages at a view change boundary in a timely fashion requires aggressive acknowledgement and retransmission.
- All core group members must transmit data regarding their current state, such as the set of other core group members to which they can actively communicate.

As the number of active members grows, installing a new view requires a larger, temporary nonlinear increase in high availability manager CPU usage. It is significantly more expensive to add or remove a single member when 50 other core group members exist, than it is to add or remove a member when 20 other members exist.

Installing a new view also triggers state changes in WebSphere Application Server components that use the high availability manager. For example, routing tables might need to be updated to reflect the started or stopped member, or a singleton service might need to be restarted on a new member.

The end result is that installing a new view results in a significant, transient spike in CPU usage. If core group sizes become too large, degenerate network timing conditions occur at the view change boundary. These conditions usually result in a failure during an attempt to install a new view. Recovery from such a failure is also CPU intensive. When insufficient CPU is available, or paging occurs, failures can quickly multiply.

Background tasks

The high availability manager periodically runs a number of background tasks, such as checking the health of highly available singleton services that it is managing. Most of these background tasks consume trivial amounts of CPU. The exceptions are the regularly scheduled discovery and Failure Detection Protocols.

The Discovery Protocol attempts to establish communications among core group members that are not currently connected, including processes that are not running. For a given core group that contains N core group members, of which M are currently running, each discovery period results in roughly M x (N – M) discovery messages. Therefore, creating a large number of processes that never start adversely affects the Discovery Protocol CPU usage.

Similarly, when the Failure Detection Protocol runs, each core group member sends heartbeats to all of its established connections to other core group members. For M active members, M x (M-1) heartbeat messages are sent. If aggressive failure detection is required, the size of the core group can adversely affect the amount of CPU usage that heartbeating between core group members consumes.

Smaller core groups positively affect the amount of CPU usage these two protocols consume. For example, if a core group contains 100 active members, 9900 heartbeat messages are sent during every failure detection period. Splitting the 100 member core group into five smaller core groups of 20 members reduces this number of message to 1900, which is a significant reduction.

External usage

Other WebSphere Application Server components, such as work load management (WLM), and on demand configuration, use high availability manager-provided services, such as live server state exchange, to maintain routing information. The amount of CPU usage that these components consume is linked to core group size. For example, the usage of the live server state exchange to build highly available routing information is linked to the size of the core group.

Distributing processes among multiple core groups

You can use two basic techniques to minimize the amount of resources that the view synchrony and related protocols consume:

- You can disable the high availability manager on processes where the services that the high availability manager provides are not used.
- You can keep core group sizes small.

The key to limiting the high availability manager CPU usage is to limit the size of the core group. Multiple small core groups are much better than one large core group. If you have large cells, create multiple core groups.

The hardware on which you are running WebSphere Application Server is also a factor in determining the core group size that is appropriate for your environment.

The current recommendation is to limit core group sizes to 50 members or so. Split large core groups into multiple, smaller core groups. If the resulting core groups need to share routing information, you can use core group bridges to bridge the core groups together.

Adjusting individual core groups based on the application mix and services used

You might need to further adjust Individual core groups based on the application mix and the high availability services that the core group members use.

- Adjust how frequently the Discovery Protocol and the Failure Detection Protocol run if the default settings are not appropriate.
- Configure the core group coordinator to run on a specific process or set of processes.
- Partition the coordinator across multiple instances if the consumption of resources by the coordinator process is noticeable.
- Configure the amount of memory that is available to the distribution and consistency services (DCS) and removable media manager (RMM) components for sending network messages when congestion is detected. Congestion can occur under some conditions, even though memory-to-memory replication is not used.

Adjusting ephemeral port ranges

The number of sockets that a core group uses is usually not a major concern. Each core group member must establish a connection with every other member of that core group. Therefore, the number of connections grows exponentially (n-squared) because each connection requires two sockets, one on each end of the connection. Because multiple machines are typically involved, normally you do not have to be concerned about the number of sockets that a core group uses. However, if you have an abnormally large number of core group members that are running on a single machine, you might have to adjust the operating system parameters that are related to ephemeral port ranges. Most operating systems have different default behavior for ephemeral port ranges.

Core group transports

Core group members communicate with each other over a specialized and dedicated network transport. Multiple transport implementations are supported, each with its own set of advantages and disadvantages.

You can use one of the following types of transports to set up communications between core group members:

- Channel framework transport, which is the default
- Unicast transport
- Multicast transport

All of these transport options require TCP/IP connections for communication between core group members. The high availability discovery protocol ensures that these connections are opened, but the selected transport opens the connections. The discovery protocol also ensures, for each core group member, that connections are opened to all other members of that core group, thereby ensuring that all running core group members are fully connected through TCP/IP.

Each core group member has a configured endpoint known as the DCS_UNICAST_ADDRESS. This endpoint contains the host and port information that indicates where the core group member is listening for TCP/IP connections.

Channel framework transport

The channel framework transport is the most flexible and scalable of the transport options and is the best option for most topologies. The channel framework transport provides the most security options for core group connections. However, the advanced features of the channel framework transport come at the cost of slightly lower performance. This performance impact is a concern only in the most demanding topologies where replication throughput is a primary objective.

The channel framework transport is the default transport type for core group member connections. If you use a channel framework transport, all communication occurs directly over the core group TCP/IP

connections. The work of opening connections and sending or receiving data is delegated to the channel framework transport and the associated channel implementations.

For example, to use a channel framework transport for distribution and consistency services (DCS) messages, you must configure either a DCS transport chain, which is the default, or a DCS_SECURE transport chain, in addition to the DCS_UNICAST_ADDRESS endpoint. A DCS transport chain uses a TCP channel for network communication. A DCS_SECURE transport chain includes both a TCP channel and an secure sockets layer (SSL) channel to add support for encrypted communication.

The channel framework function provides a common model for connection management, and contains support for implementing various channels that you can combine into a transport chain. The ability to combine various types of channels makes it possible to create custom transport chains. The channel framework function also supports port sharing, which provides additional flexibility and the potential for future customization.

If you use a channel framework transport, two different mechanisms are available to secure the core group network connections:

1. A lightweight third-party authentication (LTPA) token is used to authenticate all incoming connection requests if WebSphere Application Server administrative security is enabled.
2. Secure Sockets Layer (SSL) is used to encrypt all communications if the SSL version of a transport chain is selected.

You can use these mechanisms separately or combine them for the highest level of security possible.

Unicast transport

The communication mechanism of the unicast transport is similar to that of the channel framework transport. All communications occur over core group TCP/IP connections. The major difference is that a standard network connection, instead of a transport chain, is established and used to perform the communication between core group members.

Because a unicast transport does not go through the channel framework, this transport is somewhat faster than the channel framework transport. This performance improvement might be useful in topologies that make extensive use of memory-to-memory replication, and where the throughput that is obtained using a channel framework transport is not adequate.

Internal benchmarks show a gain in throughput using a unicast transport rather than a channel framework channel. However, this performance gain is achieved at the cost of using additional ephemeral ports.

A unicast transport has fewer security options than the channel framework transport. As with the channel framework transport, if WebSphere administrative security is enabled, an LTPA token is used to authenticate all incoming connection requests. However, no SSL encryption option is available for a unicast transport.

A unicast transport provides optimum performance with basic security. You trade the flexibility that a channel framework transport provides for improved performance. You are no longer able to do things like configure for SSL encrypted communication, or create a custom transport chain. However, in a typical application server environment, the unicast transport provides the best performance for functions like memory-to-memory session replication.

Because a unicast transport uses more ephemeral ports than a channel framework transport, it might not scale as well as a channel framework transport does as the core group size increases.

Multicast transport

The multicast transport uses IP multicast and broadcasts information to all of the other processes in the core group. Although IP multicast uses a user datagram protocol (UDP), failure detection protocol still requires TCP/IP connections to be set up between core group members.

The multicast transport requires the following additional configuration parameters:
- Multicast port, for which the default value is 23445
- Multicast IP range start, for which the default value is 239.0.0.0
- Multicast IP range end, for which the default value is 239.255.255.255

A multicast transport also provides a high level of performance. However, because a multicast transport broadcasts information to all of the members of a core group, it is best suited for situations where many core group members need the same information. If you use a multicast transport when only a few members need the same information, you risk saturating the network with unnecessary data packets.

The multicast transport is the least secure of the available core group transport types because it broadcasts information out to all members of the networks. There are no restrictions as to who could potentially receive the data or try to send data to the connection. By default, the Time to Live (TTL) field in the IP header is 1. Therefore, multicast data is restricted to the subnet on which it is sent. If all of the systems are protected and located on the same subnet, then additional levels of security might not be required.

The multicast transport is optimum when many members in the core group need to have data replicated across them. The multicast transport typically requires that all members of the core group be located on a single subnet.

Core group Discovery Protocol

When a core group member starts, no connections to other core group members exist. The task that runs the Discovery Protocol for this core group member starts as part of the core group members startup procedure. The Discovery Protocol establishes network connectivity with the other members of the core group. This task runs at regularly scheduled intervals as long as the core group member is active.

The Discovery Protocol retrieves the list of core group members and the associated network information from the WebSphere Application Server configuration settings. The Discovery Protocol then attempts to open network connections to all of the other core group members. At periodic intervals, the Discovery Protocol recalculates the set of unconnected members and attempts to open connections to those members.

When a connection is made to another core group member, the Discovery Protocol notifies the View Synchrony Protocol, and logs this event as an informational message, similar to the following message, in the SystemOut.log file.

```
DCSV1032I: DCS Stack DefaultCoreGroup at Member MyCell\anzio\nodeagent:
Connected a defined member MyCell\anzioCellManager\dmgr.
```

Connections can fail at any time for a variety of reasons. The Failure Detection Protocol detects connection failures and notifies the Discovery Protocol. The Discovery Protocol then attempts to open a new network connection to that member at the next scheduled interval.

The interval at which the Discovery Protocol runs is configurable. For Versions 6.0 and 6.0.1, the default is 15 seconds. For Version 6.0.2 and higher, the default is 30 seconds. You can use the IBM_CS_UNICAST_DISCOVERY_INTERVAL_SECS core group custom property to change this setting.

The amount of CPU cycles that the Discovery Protocol task consumes is proportional to the number of core group members that are stopped or unreachable. The CPU cycles that the Discovery Protocol task consumes is negligible at the default settings.

Core group Failure Detection Protocol

When a core group member starts, a task running the Failure Detection Protocol also starts. This task runs as long as the member is active. The Failure Detection Protocol monitors the core group network connections that the Discovery Protocol establishes. When the Failure Detection Protocol detects a failed network connection, it reports the failure to the View Synchrony Protocol and the Discovery Protocol. The View Synchrony Protocol adjusts the view to exclude the failed member. The Discovery Protocol attempts to reestablish a network connection with the failed member.

The Failure Detection Protocol uses two distinct mechanisms to find failed members:

- It looks for connections that closed because the underlying socket was closed.
- It listens for active heartbeats from the core group members.

Sockets closing

When a core group member normally stops in response to an administration command, the core group transport for that member also stops, and the socket that is associated with the transport closes. If a core group member terminates abnormally, the underlying operating system normally closes the sockets that the process opened and the socket associated with the core group transport. is closed.

For either type of termination, core group members that have an open connection to the terminated member are notified that the connection is no longer usable. The core group member that receives the socket closed notification considers the terminated member a failed member.

When a failed member is detected because of the socket closing mechanism, one or more of the following messages are logged in the SystemOut.log file for the surviving members:

```
DCSV1113W: DCS Stack DefaultCoreGroup at Member anzioCell01\anzioCellManager01\dmgr:
Suspected another member because the outgoing connection to the other member was closed.
Suspected member is anzioCell01\nettuno\ServerB. DCS logical channel is View|Ptp.

DCSV1111W: DCS Stack DefaultCoreGroup at Member anzioCell01\anzioCellManager01\dmgr:
Suspected another member because the outgoing connection from the other member was closed.
Suspected members is anzioCell01\nettuno\ServerB. DCS logical channel is Connected|Ptp.
```

The closed socket mechanism is the way that failed members are typically discovered. TCP settings in the underlying operating system, such as FIN_WAIT, affect how quickly socket closing events are received.

Active heart beating

The active heart beating mechanism is analogous to the TCP the keep alive function. At regularly scheduled intervals, each core group member sends a ping packet on every open core group connection. If the packet is acknowledged, all is assumed to be all right. If no response is received from a given member for a certain number of consecutive pings, the member is marked as failed. When a member is marked as failed, the following message is logged:

```
DCSV1112W: DCS Stack DefaultCoreGroup at Member anzioCell01\anzioCellManager01\dmgr:
Suspected member anzioCell01\nettuno\ServerB because of heartbeat timeout.
Configured Timeout is 180000 milliseconds. DCS logical channel is Connected|Ptp.
```

Active heartbeats are most useful for detecting core group members that are unreachable because the network is stopped. Active heartbeats consume some CPU usage. The amount of CPU usage that is consumed is proportional to the number of active members in the core group. The default configuration for active heartbeats is a balance of CPU usage and timely failed member detection. You can use the following core group custom properties to change the settings for active heartbeats:

- IBM_CS_FD_PERIOD_SECS, which specifies the time interval, in seconds, between consecutive heartbeats. The default value for this property is 30 seconds.

- IBM_CS_FD_CONSECUTIVE_MISSED, which specifies the consecutive number of heartbeats that must be missed before the core group member is considered failed. The default value for this property is 6.

Core group View Synchrony Protocol

The View Synchrony Protocol is established over the set of core group members that can communicate with each other. This protocol provides guaranteed, in-order message delivery for message streams that involve one sender and potentially multiple receivers. This guarantee is similar to the guarantees that TCP/IP provides for point-to-point message streams.

The set of core group members for which the View Synchrony Protocol is established is commonly referred to as a *view*. Views are unique in time and space. The act of adding or removing members from the view is called a *view change*. A view change is an important and relatively expensive synchronization point. It is also the point where synchronization, consistency and network issues are detected.

The View Synchrony Protocol is transparent to both components using the high availability manager framework and WebSphere Application Server administrators. However, disruptions in the View Synchrony Protocol might become visible, most notably when a boundary condition known as a view change occurs.

View changes

When a core group member starts, the core group transport and the associated Discovery Protocol, Failure Detection Protocol, and View Synchrony Protocol also start. The View Synchrony Protocol establishes an initial view that contains only the local member. The View Synchrony Protocol is notified when the Discovery Protocol establishes connections with other core group members. The view synchrony layer of the newly connected members then exchange state information. This information is used to determine if a new view can be formed. For example, if a newly started member discovers an existing view, it negotiates with the members of the existing view to establish a new view.

When a member of an established view stops or fails, the Failure Detection Protocol on the surviving view members detects the failure and notifies the View Synchrony Protocol. The surviving members then establish a new view that excludes the failed member.

Before a new view is established, activities that are related to the current view must be completed. All messages that are sent in the current view must be received and acknowledged by all intended recipients that are still alive. The current members must exchange a non-trivial amount of state information regarding messages sent and received. These members then perform the activities that are needed to complete the pending message activity, which might include the retransmission of messages that seem to be lost.

Installing a new view might result in significant, temporary spikes in the amount of CPU consumed and the network bandwidth used.

View change messages

A view change is a complex multipart procedure and a number of messages are logged every time a view is changed. These messages indicate the stage of view change processing that is complete or is currently running.

For example, the following message indicates that a set of core group members agreed to establish a new view and initiated the view change procedure:

```
DCSV8054I: DCS Stack DefaultCoreGroup at Member
anzioCell01\anzioCellManager01\dmgr: View change in process.
```

The following message indicates that all messages sent in the current view are completed and acknowledged:

```
DCSV2004I: DCS Stack DefaultCoreGroup at Member
anzioCell01\anzioCellManager01\dmgr: The synchronization procedure completed
successfully. The View Identifier is (2:0.anzioCell01\anzioCellManager01\dmgr).
The internal details are [0].
```

The following messages indicate that the view change completed successfully. They also specify the name or identifier for the new view, and the number of core group members in that view:

```
HMGR0218I: A new core group view has been installed. The core group is
DefaultCoreGroup. The view identifier is (3:0.anzioCell01\anzioCellManager01\dmgr).
The number of members in the new view is 2.

DCSV1033I: DCS Stack DefaultCoreGroup at Member
anzioCell01\anzioCellManager01\dmgr: Confirmed all new view members in view
identifier (3:0.anzioCell01\anzioCellManager01\dmgr). View channel type is View|Ptp.
```

The following message provides extended status regarding the state of connections and view synchrony:

```
DCSV8050I: DCS Stack DefaultCoreGroup at Member
anzioCell01\anzioCellManager01\dmgr: New view installed, identifier
(3:0.anzioCell01\anzioCellManager01\dmgr), view size is 2 (AV=2, CD=2, CN=2, DF=6)
```

In this message:

- AV is the number of core group members in the view.
- CN is the number of core group members to which this member has open connections. Normally this number is the same as the number that is specified for AV.
- CD is the number of core group members to which this member has open connections minus the number of bad members. A bad member is one that is connected to this member, but cannot currently establish a view with this member.
- DF is the number of members defined in the core group.

High availability groups

High availability groups are part of the high availability manager framework. A high availability group provides the mechanism for building a highly available component and enables the component to run in one of several different processes. A high availability group cannot extend beyond the boundaries of a core group.

A high availability group is associated with a specific component. The members of the group are the set of processes where it is possible to run that component. Therefore, a WebSphere Application Server administrator cannot directly configure or define a high availability group, and its associated set of members. Instead high availability groups are created dynamically at the request of the components that need to provide a highly available function.

Scope

A high availability group cannot extend beyond the boundaries of a core group. Therefore, a highly available component cannot fail over from a server process that is defined in one core group to a server process that is defined in a different core group.

Life cycle

Because high availability groups are dynamically created, a WebSphere Application Server administrator has no direct control over when they are created or destroyed. A high availability group is created when component code that runs in a given process calls the high availability manager framework to join a group. The calling component must provide the name of the high availability group for the high availability manager framework to join.

If a high availability group with this name does not currently exist, the high availability manager creates one, and makes this member the first member of the newly created group. If the high availability group already exists, this member is added to the set of high availability group members.

Because several different components might use the high availability manager framework, it is possible to have several different high availability groups across the same set of processes. However, each high availability group always has a unique group name.

A high availability group ceases to exist when all of the group members leave the group, which typically occurs when all of the processes that host members of a given high availability group stop.

Group name

Every high availability group has a unique name. Because any component can create a high availability group for that component to use, it is the high availability group name that ties a given component to a particular high availability group. A high availability group name is not a simple string; this name is a set of name-value pairs that the creating component specifies. A high availability group name can look like the following example:

```
Company=IBM,ComponentName=TM,policy=DefaultNoQuorumOneOfNPolicy
```

A component can specify any number of name-value pairs to create a unique name for their high availability group.

Member state

Each member of a high availability group is either idle, active or disabled. Typically, a high availability group member will be either idle or active. A member that is idle is not assigned any work, but is available as a backup if a member that is active fails. A member that is active is designated as the member to handle the component workload.

If a member is disabled, it cannot participate in the high availability group. A disabled member is not assigned any work, and is not available as a backup if an active member fails. An administrator might disable a member if they plan to remove, delete, or cycle power on the associated server. However, this action is not required.

Policy

Every high availability group has an associated policy. The policy is used to determine which members of a high availability group are active at a given point in time. The policies available for high availability groups to use are stored as part of the core group configuration. See “High availability group policies” for more information.

High availability group policies

Every high availability group has a policy associated with it. This policy is used to determine which members of a high availability group are active at a given time.

The policies that the high availability groups use are stored as part of the core group configuration. The same policy can be used by several different high availability groups, but all of the high availability groups to which it applies must be part of the same core group. Before modifying or deleting an IBM provided policy, see “High availability group policy modification guidelines” on page 367

Policy selection

Policies are statically configured, and the high availability groups that are governed by them are created dynamically. Therefore, a mechanism is required to associate a running high availability group to a configured policy. This association is accomplished by comparing the following two pieces of information.

- The high availability group name
- The policy match criteria

"High availability group policy selection process" provides more detail about how a policy is selected.

Policy settings

Some policy settings apply for all policy types while others only apply for specific policy types. Some of the policy settings also influence the overall behavior of a policy. "Implications of high availability group policy settings" on page 364 describes the various settings, the applicable policy types, and how they influence policy behavior.

Policy enforcement

Whenever one of the following conditions occurs, the high availability manager runs the policy that is associated with a high availability group and takes any appropriate action:

- A member joins or leaves that high availability group. A member leaves the group if the member fails.
- The state of a member of that high availability group changes. For example, if the state changes from idle to active, or from idle to disabled, the policy rules are reapplied.

Policy changes

The high availability manager dynamically detects policy configuration changes. Therefore, policy setting changes go into affect as soon as you save and propagate these changes. Server restarts are not required.

High availability group policy selection process

Every high availability group has a unique group name that consists of a set of name-value pairs. Every policy definition contains an attribute called *match criteria* that is also a set of name-value pairs. To determine the policy for a high availability group, the group name is compared to the match criteria of all the associated core group polices. The policy with the strongest match to the group name is assigned to the high availability group:

When selecting a policy for a high availability group, the high availability manager:

1. Finds the set of policies that are eligible to govern a high availability group. For a policy to be eligible, all name-value pairs in the match criteria of an eligible policy must be contained in the name of the high availability group.
2. Selects the policy that has the most name-value pair matches from the list of eligible policies, and uses that policy to govern the high availability group.

Any component can create a high availability group for that component to use. However, the component code must specify the name-value pairs that are used for the high availability group name. The WebSphere Application Server administrator can control the name-value pairs that make up a policy match criteria, and thereby control which policy governs a particular high availability group.

WebSphere Application Server includes a couple of predefined policies. The following examples demonstrate the matching mechanism that is used for these policies.

Clustered TM Policy

The transaction manager component uses the policy Clustered TM Policy when the component is configured for high availability. The following description illustrates why, under these conditions, this policy is selected for the transaction manager high availability group:

- A cluster member process, such as ServerA, is started.
- The transaction manager component code joins a high availability manager to the high availability group named:

  `GN_PS=testCell\testNode\ServerA,IBM_hc=MyCluster,type=WAS_TRANSACTIONS`
- ServerA is defined as a member of the DefaultCoreGroup core group for which the following policies are defined:
  - Clustered TM Policy, which has the match criteria type=WAS_TRANSACTIONS.
  - Default SIBus Policy, which has the match criteria type=WSAF_SIB.
- The high availability manager compares the group name to the match criteria for the two available policies. The high availability manager eliminates the Default SIBus Policy because the match criteria is not a proper subset of the high availability group name. The high availability manager determines that Clustered TM Policy is the closest match because:
  1. The match criteria for that policy includes the name-value pair type=WAS_TRANSACTIONS, which is also specified in the high availability group name. Therefore, the match criteria is a proper subset of the high availability group name.
  2. The match criteria for that policy more matches (one) than the match criteria for Default SIBus Policy, which is eliminated because it does not have any matches.

Administrator TM Policy

This example builds on the previous example and demonstrates how an administrator can define a new policy to govern the transaction manager high availability group. In this example the same high availability group name and default policies that are described in the previous example are used. However, in this example, the administrator creates a new policy in the DefaultCoreGroup configuration called the Administrator TM Policy. For the high availability manager to select this new policy, the policy must be eligible and contain more matches than any other policy.

The following description illustrates why, under these conditions, the policy Administrator TM Policy is selected for the transaction manager high availability group:

- The cluster member process ServerA is started.
- The transaction manager component code joins a high availability manager to the high availability group named:

  `GN_PS=testCell\testNode\ServerA,IBM_hc=MyCluster,type=WAS_TRANSACTIONS`
- ServerA is defined as a member of the DefaultCoreGroup core group, for which the following policies are defined:
  - Clustered TM Policy, which has the match criteria type=WAS_TRANSACTIONS.
  - Default SIBus Policy, which has the match criteria type=WSAF_SIB.
  - Administrator TM Policy, which has the match criteria IBM_hc=MyCluster,type=WAS_TRANSACTIONS.
- The high availability manager compares the group name to the match criteria for the available policies. The high availability manager eliminates the Default SIBus Policy because the match criteria is not a proper subset of the high availability group name. It determines that Clustered TM Policy and Administrator TM Policy are both eligible policies, because their match criteria are proper subsets of the high availability group name:
  - Clustered TM Policy contains the name-value pair type=WAS_TRANSACTIONS, which is also specified in the high availability group name.
  - Administrator TM Policy contains the name-value pairs IBM_hc=MyCluster and type=WAS_TRANSACTIONS, which are both specified in the high availability group name.

Because Administrator TM Policy has two matching pairs, IBM_hc=MyCluster and type=WAS_TRANSACTIONS, and Clustered TM Policy has only one matching pair, type=WAS_TRANSACTIONS, the high availability manager associates Administrator TM Policy with the transaction manager high availability group.

Ambiguous Matches

Do not configure identical match criteria for multiple policies in the same core group. Configuring identical match criteria causes an ambiguous match to the associated high availability group. Because a high availability group can only be associated with one policy, if the previously described matching mechanism does not result in a single policy match, the high availability manager puts the high availability group in error state, and does not make any of the group members active. Depending on the nature of the problem, the high availability manager might write one of the following error messages to the SystemOut.log file:

```
HMGR0301W: No policy was located for the group named {0}
HMGR0302W: Multiple policies match the group named {0}, Matching Policies are {1}
```

You can use the administrative console to view the policies associated with a high availability group and the current state of members of that group.

Implications of high availability group policy settings

All of the settings that are specified for a policy affect how the high availability manager governs a high availability group associated with that policy. Some of the policy settings are policy type specific, while others apply to all policy types. It is important to understand the implications for all of the associated high availability groups before you change the settings of an existing policy.

Implications of the Policy type setting

The policy type determines which members of a high availability group are automatically made active when the servers containing these members start. You can not directly change the policy type of an existing high availability group policy. If you need to change the policy type, you must create a new policy with a different policy type and give it a match criteria that makes the high availability manager select a new policy instead of the original one to associate with the high availability group.

Before creating a new policy with a different policy type, you must determine which components are using the high availability groups that are governed by the original policy and make sure that those components support the new policy type. For example, the service integration bus (SIB) component might require a One of N policy for its high availability group, because it only wants one group member active at a given ime. If you change the policy that is associated with the service integration bus high availability group to be an All Active policy, the service integration bus high availability support might not function properly and data corruption can occur.

You can select one of the following policy types when you create a new policy:

All active policy
When this policy is selected, all of the members of the high availability group are made active.

M of N policy
When this policy is selected for a high availability group with N members, M of them are made active. The number that M represents is configurable in the policy settings. You can use the Preferred servers setting to designate which members of the group to activate first.

No operation policy
When this policy is selected, none of the high availability group members are made active. You can use the administrative console to manually activate specific group members.

One of N policy
When this policy is selected for a high availability group with N members, only one member of the group is made active. You can use the Preferred servers setting to designate which member of the group are made active.

Static policy
When this policy is selected, only the members specified in the Static group servers setting are made active.

Implications of the Preferred servers setting

With the One of N and M of N policy types, you can set up a list of preferred servers as part of the policy settings. The preferred server list enables an administrator to indicate a preference as to which high availability group member is made active. If no preferred server list is specified, any of the available high availability group members can be selected as the member to activate. If a preferred server list is specified, then the member to activate is selected from this list, in order of preference. The most preferred server is the first one on the list. The following example demonstrates how a policy uses of the preferred server list.

Example:

A high availability group has three members that are located on application servers named ServerA, ServerB, and ServerC. This group is governed by a One of N policy, under which only one of the three members can be active at a given time. When all three members are running and available at the time that the policy is enforced:

- If no preferred servers are specified, the high availability manager randomly selects one of the three members and makes it active.
- If ServerB is the only server on the preferred servers list, the high availability manager makes the member located on this server active before either of the other two members, provided the member located on this server is available when the policy is enforced.
- If all three of the application servers are listed on the preferred servers list in the following order, and if all other things are equal, the high availability manager makes the member that is located on ServerC active:

  ServerC

  ServerA

  ServerB

The two other policy settings that directly affect how the preferred server list is used are the Failback and Preferred servers only settings.

Implications of the Failback setting

The Failback setting is used to specify what happens to the high availability group member on the most preferred server when it is restarted following a failure. The affect of the Failback setting on a member is best demonstrated with two examples.

During startup example:

A high availability group has three members that are located on application servers named ServerA, ServerB, and ServerC. This group is governed by a One of N policy, under which only one of the three members can be active at a given time. The server named ServerB is the only server on the preferred server list. In this example, none of the servers are started.

When ServerA starts, the One of N policy dictates that the high availability manager makes a member active. Because this application server is the only server running, the member on ServerA is activated. When we start ServerB, which is the only server on the preferred server list, one of two things happens:

- If Failback is enabled when ServerB starts, the high availability manager deactivates the currently active member and activates the member on ServerB, because ServerB is on the preferred server list.
- If Failback is disabled when ServerB starts, the currently active member remains the active member.

Following a failure example:

A high availability group has three members that are located on application servers named ServerA, ServerB, and ServerC. This group is governed by a One of N policy, under which only one of the three members can be active at a given time. ServerB is the only server on the preferred server list and is the only member that is currently active. If ServerB fails, the high availability manager activates one of the remaining members to replace that member. The Failback setting determines what happens after ServerB is repaired and restarted.

- If Failback is enabled when ServerB restarts, the currently active member is deactivated, and the member on ServerB is activated, because ServerB is still the most preferred server
- If Failback disabled when ServerB restarts, the currently active member remains the active member.

Implications of the Preferred servers only setting

The Preferred Servers Only setting is used to instruct the policy to activate members on preferred servers only. With this setting enabled, only members running on the servers that are specified in the preferred servers list are activated. If no preferred servers are specified, or no preferred servers are currently available, then no members are activated.

During startup example:

A high availability group has three members that are located on application servers named ServerA, ServerB, and ServerC. This group is governed by a One of N policy, under which only one of the three members can be active at a given time. ServerB the only server on the preferred server list. In this example, none of the servers are started.

When ServerA starts, the One of N policy dictates that the high availability manager activates a member. Because ServerA is the only server running, the member on ServerA is activated. Because it is the only server on the preferred server list, when ServerB starts, one of two things happens:

- If Preferred servers only is enabled when ServerA or ServerC starts, no member is activated because the high availability manager can only activate a member that is located on a server that is on the preferred servers list. When ServerB starts, the high availability manager activates the member on ServerB because ServerB is on the preferred servers list.
- If Preferred servers only is disabled when ServerA starts, the member on ServerA is activated because any member of the group can be the active member. When ServerB or ServerC starts, no activation occurs because the member on ServerA is already active.

Following a failure example:

A high availability group has three members that are located on application servers named ServerA, ServerB, and ServerC. This group is governed by a One of N policy, under which only one of the three members can be active at a given time. ServerB the only server on the preferred server list. The member located on ServerB is the active member. If ServerB fails, one of two things happens:

- If Preferred servers only is enabled when ServerB fails, the high availability manager can only activate another members that is located on a server that is included on the preferred servers list. Because ServerB is the only server on the preferred servers list, no other member is activated.

- If Preferred servers only is disabled when ServerB fails, the high availability manager activates one of the remaining members to replace the member on ServerB.

Implications of the Static group servers setting

You can specify a list of static group servers as part of the configuration settings for a static policy type. When a high availability group is governed by a static policy type, the static group server list defines which group members are activated if it is possible to do so.

Implications of the Is alive timer setting

The Is alive timer setting controls how frequently the high availability manager checks the health of the active group members that are governed by a given policy. The high availability manager can detect two fundamentally different kinds of failures:

- It can detect when an entire process stops functioning or terminates. This type of failure detection does not depend on the value that is specified for the Is alive timer setting.
- It can detect when a program fails. This type of failure detection does depend on the value that is specified for the Is alive timer setting. The value that is specified for the Is alive time setting determines the amount of time that might pass before a processing problem that does not cause the entire process to stop functioning or terminate is detected.

The administrator has the ability to specify the Is alive timer at the policy level, where it applies to all the members that are governed by this policy, at the process level where it applies to all members running in a particular process. The administrator can also disable this type of failure detection at either of these levels.

Implications of the Quorum setting

Quorum is a mechanism that you can use to protect resources that, in the event of a failure, are shared across members of a high availability group. When enabled, the policy does not activate any group members until quorum is achieved. A high availability group does not achieve quorum until a majority of the members are running. For example, if there are n members in a group, (n/2) + 1 servers must be online to achieve quorum.

Quorum is an advanced function that is designed to work with clusters, specialized component code, and a hardware control facility. Currently, none of the high availability groups supporting WebSphere Application Server components use the quorum mechanism. Therefore, do not enabled the Quorum setting.

High availability group policy modification guidelines

Use the following guidelines to help determine when to create a new high availability group policy, and when to modify or delete an existing policy.

Do not delete the default IBM provided policies

If you want to override one of the default policies that IBM provides, it is recommended that you do not delete the current policy. Instead, create a new policy with more specific match criteria. The policy with the greatest number of matches is the one that is used. Not deleting the IBM provided policy enables you to revert back to that policy if a problem occurs with your new policy.

Do not try to change the type of an existing policy

After a policy for a high availability group is created, you can change some of the policy attributes such as preferred servers, or failback, but you cannot change the policy type. If you need to change the policy type, you must create a new policy and then use match criteria to associate it with the appropriate high availability group. See “High availability group policy selection process” on page 362 for a description of how the high availability manager selects a policy for a high availability group.

Make sure that you know which policies a component using that high availability group supports before creating a new policy to associate with that group

A component does not necessarily support all policy types and options. Therefore, before changing the policy that is associated with a given high availability group, make sure you fully understand if the application server code using that high availability group supports the change. For example, if you want to change the type of policy that is associated with the high availability group used by the transaction manager component, make sure the transaction manager code supports the new policy type before making the change.

Do not use the same match criteria for multiple policies in the same core group

If you have multiple policies configured with identical match criteria, the policy match to the associated high availability group is ambiguous. If you are creating a new policy to replace an older policy that you created, you might need to delete the older policy to specify the appropriate match criteria. Another alternative is to specify additional match criteria in each policy so no ambiguity exists as to which policy is controlling the high availability group.

High availability data sources

Without continuous access to the application data, your application servers cannot process client requests. WebSphere Application Server supports various methods of maintaining data availability.

If your data resides on i5/OS servers, you can use one of the following options to achieve higher availability for your data:

Data replication

To ensure the availability of all components of your application server environment, you can use data replication to create a backup copy of your application data. Data replication uses clustering, remote journaling, and third-party data replication software to maintain two physically separate copies of application data. Data replication can provide disaster recovery capabilities for the databases because the systems in the cluster can be geographically separated.

Clustering provides the basis for communication between two or more servers. This communication is necessary to back up your data on a physically separate machine.

Remote journaling creates a copy of your application data to assure hot backup. There are two types of remote journaling:

- Synchronous remote journaling, where data is written to both the primary and backup databases simultaneously. This type of journaling ensures that no entries are lost in the event of a system failure. However, it can negatively impact performance, because it requires the application to wait while data is written to both databases.
- Asynchronous remote journaling, where data is written to the primary database directly from the application. This type of journaling, enables the application to continue processing client requests while data is copied to the backup system. It does not impact application performance, but the most recent entries might be lost if a failure occurs.

Switchable disk

Switchable disk uses clustering, local journaling, and Independent Auxiliary Storage Pool (IASP) to ensure the availability of application data. The application data is stored in an IASP, and if a failure occurs, the IASP can be switched to another node.

The advantage of switchable disk is that no data replication occurs. Therefore, you do not have to worry about data synchronization. However, the database becomes a single point of failure. In addition, the backup node must be physically close to the primary node. Therefore, this configuration does not provide disaster recovery.

Switchable disk uses the following components and services:

- **A server cluster**. You must have the High Availability Switchable Resources function (5722-SS1 Option 41) installed on all of the nodes in your cluster.
- **Clustering**. When you use switchable disk, clustering provides:
  - The services required to switch the IASP.
  - Wellness monitoring of backend database partitions.
- **Local journaling**. Switchable disk uses local journaling to preserve database transaction boundaries.

Switchable disk is best utilized in a multiple cell topology. that includes these features:

- All of the cells process client requests as described in multiple cell topology.
- One of the cells hosts the primary database system, which receives data from application servers on the other cells.
- The primary database system connects to an IASP.
- One of the other cells hosts the secondary database system, which is inactive unless there is a failure in the primary system. If the primary system fails, application data is directed to the secondary database system in this other cell, and the IASP connects to the secondary system.

Changing the number of core group coordinators

For a large cell, multiple coordinators are recommended.

Before you change the number of core group coordinators for a cell:

- Make sure that you are familiar with the content of the “Core group coordinator” on page 350 topic.
- Determine the number of coordinators for the core group. A general guideline is approximately one coordinator per 20 core group members.

Change the number of core group coordinators only if:

- IBM support instructs you to do so.
- You are directed to do so as part of the installation process for another WebSphere product.
- The coordinator process is consuming abnormally high amounts of memory or CPU resources.
- You are scaling up the number of servers in a cell.

To change the number of core group coordinators for a core group:

1. In the administrative console, click **Servers > Core groups > Core group settings** and select an existing core group.
2. In the **Number of coordinators** field, specify the number of coordinators that you want for this core group.
3. Click **OK**, and then click **Save**.
4. Click **Synchronize changes with nodes** and then click **Save** again to save your changes.
5. If you need to change the list of preferred coordinator servers, click **Preferred coordinator servers** and then add or delete the appropriate application servers from the list of preferred coordinator servers.
6. Click **OK** and then click**Review**.
7. Select **Synchronize changes with nodes**, and then click **Save**.

Your changes take effect immediately after synchronization completes. The members of the core group pick up these changes dynamically.

Core group settings

Use this page to create a core group or to edit an existing core group. A core group is a component of the high availability manager function. It can contain standalone servers, cluster members, node agents and the deployment manager. A core group must contain at least one node agent or the deployment manager.

Before you create a core group you must understand the relationship of core groups in a high availability environment and know how you intend to use each core group.

To view this administrative console page, click **Servers > Core groups > Core group settings > New** or *Select an existing core group for editing*.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information.

After you specify your core group settings, click **Apply** before defining additional properties or setting up a core group bridge.

Extended information about the core group fields:

Name

Specifies the name of the core group. This field can only be edited when you create new core groups.

If you are defining a new core group, specify a name that is unique among the existing core groups. It is helpful to other WebSphere Application Server administrators if the name helps define the use of this core group and if it is consistent with the names of the other core groups in the cell.

This field can contain alpha and numeric characters. The following characters cannot be used in this field:
`# \ / , : ; " * ? < > | = + & % '`

Also, the name cannot begin with a period (**.**) or a blank space. A blank space does not generate an error. However, leading and trailing blank spaces are automatically deleted.

For example, `DefaultCoreGroup` is the name of the core group that contains the deployment manager server process.

Description

Specifies a description of the core group. In environments where there are multiple system administrators this field can help these administrators understand the overall organization of the core groups. The supported length of this field is quite large. However, long descriptions take time to load and can cause a delay when displaying the page.

Example: ″Default Core Group. The default core group cannot be deleted.″ is the description of the DefaultCoreGroup.

Number of coordinators

Specifies the number of coordinators for this core group. The coordinator is the aggregation point for high availability manager information. The coordinator determines group membership and communicates state and status to the other members of the core group.

The default value is one coordinator, although multiple coordinators are advisable for large core groups. All of the group data must fit in the memory of the allocated coordinators. One coordinator can run out of memory in a system with a large core group, which can cause the system to work improperly.

Transport type

Specifies the transport mechanism to use for communication between members of a core group. This is a required field.

Channel framework
Channel framework is the default transport type. It uses the channel framework service to incorporate port reusability and shared port technology into the communication system.

Unicast
Unicast is a targeted network model that focuses on a direct recipient for communication. This type of communication is most suitable when the intended message is sent to a specific set of recipients.

Multicast
Multicast consists of a broadcast network model. This model broadcasts communication across the defined network, depending upon the values that are provided for the multicast settings. Multicast settings are suitable when there are many recipients for the intended message; otherwise broadcast communication tends to overload the network with traffic, and can impact performance goals.

Important: If a core group needs to use the core group bridge service, you must select Channel framework as the transport mechanism for that core group. If you select Unicast or Multicast, you might receive error message CHFW0029E, which indicates that the transport chain could not be initialized because the address was already in use.

Channel chain name

Specifies the name of the channel chain if you select channel framework for the transport type.

Multicast settings

Specifies the following settings for a multicast transport type. These settings are only valid if you select multicast for the transport type:

- **Multicast port**

  The port setting tells the coordinator where to scan for transmissions. When setting this value, verify that you are specifying a port that is not used by another network communication device. Setting a port value that has conflicts causes problems with your high availability manager infrastructure.
- **Multicast group IP start**

  Specify the starting Internet Protocol (IP) address of the intended communication area.
- **Multicast group IP end**

  Specify the ending IP address of the intended communication area. Plan the network to accommodate scalability.

Additional properties

Core group servers
Specifies the server processes that belong to the core group. Server processes include the deployment manager, node agents, application servers, and cluster members. You can use the panel that displays to move server processes to a different core group.

Custom properties
Specifies the custom properties that are used for configuration purposes.

Policies
Use to define the policies that determine which members of a high availability group are made active.

Preferred coordinator servers
Specifies which core group servers are preferred coordinator servers.

Related topics

Core group bridge settings

Click on to specify core group bridge communication settings between core groups.

Group name properties

Specifies one or more name=value pairs as the match criterion for a high availability group. If you specify more than one name=value pair, use a comma to separate the pairs. You can specify an asterisk (*) to obtain the selected information for all of the high availability groups within this core group.

When a WebSphere Application Server component creates a new high availability group, it establishes a map of that group's properties as the group name. This map is used to uniquely identify that high availability group.

After you specify a match criterion or an asterisk:

- Select **Calculate** to determine how many high availability groups have names that match this match criterion.
- Select **Show groups** to view a list of the currently running high availability groups that match this match criterion. For each group, this list indicates:
  - Its high availability group name
  - Whether or not quorum has been enabled
  - The policy that is associated with the high availability group. If more than one policy is listed for a high availability group, you must change the match criterion for one or more of your policies so that only one policy is associated with this high availability group.
  - Its status (either the OK icon or the Error icon). If only one policy is listed in the Policy column, the OK icon is displayed in the Status column. If more than one policy is listed Policy column, the Error icon is displayed in the Status column.
- Select **Show severs** to view a list of servers which are hosting active members of the high availability groups that match the specified Group name properties. For each server, this list indicates:
  - The names of the servers which are hosting the active high availability group members.
  - The name of the node on which these servers resides.
  - The version of the WebSphere Application Server product on which these servers are running.
  - The number of high availability group members that are currently active on these servers.

Example: Suppose the following high availability groups are defined for a core group:

- Component A uses the following properties for its group name: [name=compA, policy=oneofN, owner=smith]
- Component B uses the following properties for its group name: [name=compB, policy=MofN, owner=smith]
- Component C uses the following properties for its group name: [name=compC, policy=oneofN, owner=smith]

If you specify `policy=oneofN` in the **Group name properties** field and then select **Show groups**, the groups for components A and C are listed.

If you specify `owner=smith` in the **Group name properties** field and then select **Show groups**, the groups for components A, B and C are listed.

If you specify all of component C's name properties in the **Group name properties** field:

```
name=compC,policy=oneofN,owner=smith
```

Then select **Show groups**, only the group for component C is listed. Note that the properties are separated by commas. There are no blank spaces.

Core group custom properties

Use these custom properties for advanced configurations for core groups.

To configure a custom property, complete the following steps:

1. In the administrative console, click **Servers > Core groups**, and then select one of the listed core groups.
2. Under Additional Properties, click **Custom properties**
3. If the custom property is in the list of defined custom properties, click on that property and then enter an appropriate value in the Value field.

   If the custom property is not in the list of defined custom properties, click **New** and then enter the name of the custom property in the Name field and an appropriate value in the Value field.

IBM_CS_LS_DATASTACK_MEG

Use this custom property to eliminate a condition that is reported by a message that is displayed repeatedly in your `SystemOut.log` file.

You might see a message, similar to the following message, repeated multiple times in the SystemOut.log file:

```
[9/24/04 13:28:19:497 CDT] 00000013 VSync         W
DCSV2005W: DCS Stack DefaultAccessPointGroup.P at Member 172.16.1.86:9353:
The amount of memory available for synchronization is low. The configured memory
size is 418816 bytes. Currently used memory size is 420307 bytes.
```

If the member IP address is in the format of a dotted decimal IP address and port, you can eliminate these messages by increasing the amount of memory that is allocated to the core stack that is used for core group communication. Increase the value of this property until you no longer see the message in your SystemOut.log file. Because the memory is dynamically allocated, setting a larger stack size than you need does not cause memory problems.

Note: You can also set this custom property:

- On the core group bridge interface that contains the particular core group member that is in the messages.
- On the access point group or the core group access point for the particular core group member that is in the messages.

See “Core group bridge custom properties” on page 426 for more information about how to set the property at those levels.

Units	megabytes
Default	5

IBM_CS_IP_REFRESH_MINUTES

Use this custom property to adjust how frequently the core group IP cache is cleared.

The caching of name-to-IP address information at the core group level eliminates some of the system overhead required to assign IP addresses to core group members.

Units	seconds
Default	60
Acceptable values	Any positive integer. 1 is the minimum value that can be specified

IBM_CS_UNICAST_DISCOVERY_INTERVAL_SECS

Use this custom property to change how frequently the high availability manager Discovery Protocol checks for new core group members. A new core group member is not able to communicate with other core group members until the Discovery Protocol establishes communication between the new member and the existing members.

See “Core group Discovery Protocol” on page 357 for more information about the Discovery Protocol.

Units	seconds
Default	30 seconds.

IBM_CS_FD_PERIOD_SECS

Use this custom property to change how frequently the Failure Detection Protocol checks the core group network connections that the discovery protocol establishes. The Failure Detection Protocol notifies the Discovery Protocol if a connection failure occurs.

See “Core group Failure Detection Protocol” on page 358 for more information about the Failure Detection Protocol.

Units	seconds
Default	6

IBM_CS_SOCKET_BUFFER_SIZE

Use this custom property to change the size of the socket buffer that the core group transport obtains.

“Configuring core group socket buffers” on page 383 includes a table that shows the relationship between the values that can be specified for this property and the underlying memory allocation size per socket buffer type.

Units	One of the following: • No over rides • small • medium • large
Default	2 megabytes for all buffer types.

Configuring core group preferred coordinators

The high availability manager uses an ordered list of preferred core group servers when it selects servers to host the coordinators. If the default list is inappropriate, you can change the list such that other servers are selected as coordinators.

Use the following guidelines to determine the appropriate ordered list of preferred coordinators:

- Make sure that you are familiar with the content of the “Core group coordinator” on page 350 topic.
- Select a set of servers that infrequently start and stop, are stable, and that are located on capable systems with large amounts of system memory and a fast processor.

You might want to change the ordered list of preferred core group servers if:

- The system where the default coordinator is located has insufficient resources or capacity to serve as a coordinator.
- A server that normally gets selected as the coordinator starts and stops frequently.

To change the ordered list of preferred core group servers:

1. In the administrative console, click **Servers > Core groups > Core group settings** and select an existing core group.
2. Click **Preferred coordinator servers** and then add or delete the appropriate application servers from the list of preferred coordinator servers. Use **Add** and **Remove** to move servers into and out of the list of core group servers on which the coordinator service can run. Use **Move up** and **Move down** to adjust the order of the servers within this list. Make sure that the most preferred server is at the beginning of the list and the least preferred server is at the end.
3. Click **OK** and then click**Review**.
4. Select **Synchronize changes with nodes**, and then click **Save**.

Your changes take effect immediately after synchronization completes. The members of the core group pick up these changes dynamically.

Preferred coordinator servers settings

Use this page to define an ordered list of preferred core group servers. This list indicates where the high availability manager coordinators will run.

To view this administrative console page, click **Servers > Core groups > Core group settings > New >** or *select an existing core group* **> Preferred coordinator servers**.

Use **Add** and **Remove** to move servers into and out of the list of core group servers on which coordinator service will run. Use **Move up** and **Move down** to adjust the order within this list. Make sure that the most preferred server is at the top of the list and the least preferred server is at the bottom.

Click **OK** to make your changes effective. Click **Save** to save and synchronize your changes with all managed nodes.

Configuring a core group transport

When you configure a core group, you can specify the type of network transport that you want the high availability manager to use for network communications.

You should do the following before you start to configure a core group transport:

- Make sure that you are familiar with the content of the “Core group transports” on page 355 topic.
- Confirm that all node agents in the core group are running.
- Determine which transport type is appropriate for the core group.

You need to change the transport setting for a core group if the current transport type does not meet your needs. For example, you might need to increase security, you might want to maximize replication throughput, or a core group might need to use the core group bridge service.

Important: If a core group needs to use the core group bridge service, you must select Channel framework as its transport mechanism. If you select Unicast or Multicast, you might receive error message CHFW0029E, which indicates that the transport chain could not be initialized because the address was already in use.

To change the core group transport for a core group:

1. In the administrative console, click **Servers > Core groups > Core group settings** and select an existing core group or click **New** if you are creating a new core group.
2. Under Transport type, select the type of transport that you want to use for this core group. You can select one of the following types of transports:

- Channel framework, which is the default transport type. Channel framework is a flexible point-to-point transport. If you choose this transport type, you must also select one of the transport chains that is listed in the Channel chain name field.
  - If you select DCS, the high availability manager performs related network communication over non-secure sockets.
  - If you select DCS-Secure, the high availability manager performs related network communication over Secure Sockets Layer (SSL) encrypted sockets.
- Unicast, which is a point-to-point transport over standard non-secure operating system-level TCP/IP network connection facilities.
- Multicast, which is an IP multicast transport. This type of transport broadcasts all information to all of the other processes in the core group. If you choose this transport type, you must also specify:
  - In the Multicast port field, the port number for this multicast transport. The default port is 23445.
  - In the Multicast group IP start field, the first IP address in the range of IP addresses to use for this multicast transport . The default IP address is 239.0.0.0
  - In the Multicast group IP end field, the last IP address in the range of IP addresses to use for this multicast transport . The default IP address is 239.255.255.255.

3. Click **OK**, and then click **Save**.
4. Select **Synchronize changes with nodes**, and then click **Save** again.
5. Restart the servers containing core group members.

After the servers complete their restart, they all use the new transport.

Interoperating with Version 6.0.1.2 processes

The high availability manager supports multihomed hosts, which means that WebSphere Application Server processes can communicate with each other even if they are running on different versions of the product. If you are running Version 6.0.1.2 processes that need to communicate with each other, there are high availability manager-related issues that you need to consider.

Know the version levels of the processes that need to communicate with each other. If no V6.0.1.2 processes exist, see “Interoperating with Version 6.0.2 and later processes” on page 378.

In Version 6.0.1 without Fix Pack 2 installed, an acceptable value for the DCS_UNICAST_ADDRESS host name field is either a host name, for example, myhost.mydomain, or a textual IP address, for example, 192.168.0.2. If a host name is specified, Domain Name Services (DNS) is used to resolve the host name to an IP address. If the host supports multiple IP addresses, where the host name has multiple mappings in DNS, a host name is ambiguous and a textual IP address is required.

After the installation of Version 6.0.1 Fix Pack 2, the asterisk is recognized as a valid value for the DCS_UNICAST_ADDRESS host name field. When this field is set to an asterisk, multihome support allows the high availability manager to open and receive connections on all IP addresses available for the host.

With the introduction of Version 6.0.1 Fix Pack 2, processes can be classified into three separate categories:

Type 1
: Processes that can operate only in single-IP mode. This category includes processes on Version 6.0 nodes or Version 6.0.1 nodes that do not have Version 6.0.1 Fix Pack 2 installed.

Type 2
: Processes on Version 6.0.1 nodes that have Version 6.0.1 Fix Pack 2 installed, but do not have the DCS_UNICAST_ADDRESS host name field for the process set to an asterisk.

Type 3
: Processes on Version 6.0.1 nodes that have Version 6.0.1 Fix Pack 2 installed and have the DCS_UNICAST_ADDRESS host name field for the process set to an asterisk.

The following interoperability rules apply to these process types:

- Type 1 and type 2 processes can interoperate.
- Type 2 and type 3 processes can also interoperate.
- Type 1 and type 3 processes cannot interoperate.

  Therefore, careful planning is necessary before converting processes to type 3 processes.

To enable interoperating with Version 6.0.1.2 processes:

1. Add multihome capability by installing Version 6.0.1 Fix Pack 2. This step changes the existing type 1 processes to type 2 processes.
   a. Apply Version 6.0.1 Fix Pack 2 to all existing nodes. Nodes on which Version 6.0.1 Fix Pack 2 is installed continue to interoperate with nodes on which this fix pack is not installed, as long as the configuration does not change.
2. Stop all application servers.
3. Ensure that the deployment manager and all of the node agents are running.
4. Change the DCS_UNICAST_ADDRESS Host name field of each process to an asterisk. This step changes the existing type 2 processes to type 3 processes.
   a. In the administrative console, go to the configuration data for one of the processes on the new nodes:
      - For an application server, click **Servers > Application Servers >** *server_name*, and then, under Additional Properties, click **> Ports >** *port_name* .
      - For a node agent, click **System administration > Node agents >** *node_name*, and then, under Additional Properties, click **> Ports >** *port_name* .
      - For a deployment agent, click **System administration > Deployment manager**, and then, under Additional Properties, click **> Ports >** *port_name* .
   b. Click **View associated transports** for the port that is associated with the DCS transport channel that you want to review.
   c. Click **DCS_UNICAST_ADDRESS**, and enter the name of the new host in the Host field.
   d. Click **OK**, and then click **Save**.
   e. Repeat the previous steps until the new host name is added for all of the processes on the new nodes.
   f. Select **Synchronize changes with nodes**, and then click **Save** again.
5. Stop all of the servers that contain the processes with configuration changes.
6. Restart these servers.

All of the processes can communicate with each other.

After multihome support is enabled, all of the new nodes that are added to the installation must have multihome support enabled. The base Version 6.0.1 code and Version 6.0.1 Fix Pack 2 must be installed before profiles are created and the node is added. If this procedure is not observed, manual configuration is required to enable the processes to connect.

Your configuration cannot contain a mix of type 1 and type 3 processes. To ensure a valid configuration:

1. Check for type 1 processes. Type 1 processes log the following message, which indicates that the host name for another process in the core group is an asterisk:

   ```
   HMGR0024W: An error was encountered while looking up the IP address for
   the host name of a core group member. The host name is * and the server
   name is myCell01\myCellManager01\dmgr. The member will be excluded from
   the core group.
   ```

2. Check for type 3 processes. Type 3 processes do not display in a view with type 1 processes, but can be detected by examining the HMGR0218 messages the various processes log. If the processes connect, the same message is logged across all processes. Specifically, processes that connect have the same view identifier and the same number of processes in the view.

```
HMGR0218I: A new core group view has been installed. The core group is
defaultCoreGroup. The view identifier is (8:0.spoletoCell01\
spoletoCellManager01\dmgr). The number of members in the new view is 2.
```

Interoperating with Version 6.0.2 and later processes

The high availability manager supports multihomed hosts, which means that WebSphere Application Server processes can communicate with each other even if they are running on different versions of the product. If you are running Version 6.0.2 and later processes that need to communicate with each other, you need to consider there are high availability manager-related issues.

Know the version levels of the processes that need to communicate with each other. If Version 6.0.1.2 processes exist, see “Interoperating with Version 6.0.1.2 processes” on page 376 for additional considerations.

When processes are created on Version 6.0.2 or later nodes, the host name field in the DCS_UNICAST_ADDRESS endpoint is set to an asterisk. When you use an asterisk in the host name field, multihome support allows the high availability manager to open and accept connections using any IP address that is valid for that machine.

Special considerations are required when Version 6.0.2 nodes interoperate with Version 6.0 and Version 6.0.1 nodes without Fix Pack 2 (6.0.1.2), because these earlier versions do not contain high availability manager multihome support. Version 6.0 and Version 6.0.1 processes do not recognize an asterisk as a valid value in the DCS_UNICAST_ADDRESS host name field. Therefore, these processes cannot connect to Version 6.0.2 processes that are configured with an asterisk in the DCS_UNICAST_ADDRESS host name field. When Version 6.0.2 processes must interoperate with Version 6.0 or Version 6.0.1 processes, the DCS_UNICAST_ADDRESS for all Version 6.0.2 processes must be configured to use a value other than an asterisk. This configuration is done by setting the host name field in the DCS_UNICAST_ADDRESS endpoint to a host name or to a textual IP address.

- If the host is configured to use a single IP address, a string host name, for example, myhost.mydomain, is sufficient.
- If the host is configured to use multiple IP addresses, then a textual IP address, for example, 192.168.0.2, is required.

As you add Version 6.0.2 nodes and create servers into a mixed-release cell, you must set the host name field of the DCS_UNICAST_ADDRESS for all processes on the new nodes, including the node agent, to one of the two values previously specified. You must then restart the Version 6.0.2 processes to pick up the new value.

To change the host name field for a process:

1. In the administrative console, go to the configuration data for one of the processes on the new nodes:
   - For an application server, click **Servers** > **Application Servers** > *server_name*, and then, under Additional Properties, click **>** **Ports** > *port_name* .
   - For a node agent, click **System administration** > **Node agents** > *node_name*, and then, under Additional Properties, click **>** **Ports** > *port_name* .
   - For a deployment agent, click **System administration** > **Deployment manager**, and then, under Additional Properties, click **>** **Ports** > *port_name* .
2. Click **View associated transports** for the port that is associated with the DCS transport channel you want to review.
3. Click **DCS_UNICAST_ADDRESS**, and enter the name of the new host in the Host field.
4. Click **OK**, and then click **Save**.
5. Repeat the previous steps until the new host name is added for all of the processes on the new nodes.
6. Select **Synchronize changes with nodes**, and then click **Save** again.
7. Stop all of the servers that contain the processes with configuration changes.

8. Restart these servers.

All of the processes can communicate with each other.

If Version 6.0.2 processes are not configured properly, the Version 6.0 and Version 6.0.1 processes might not start or might not be able to connect to the Version 6.0.2 processes.

The condition can be detected in one of the following ways:

- Version 6.0 and Version 6.0.1 processes log the following message:

  ```
  HMGR0024W: An error was encountered while looking up the IP address for
  the host name of a core group member. The host name is * and the server
  name is myCell01\myCellManager01\dmgr. The member is excluded from the
  core group.
  ```

  This message indicates that the host name for another process in the core group is an asterisk.
- The Version 6.0, Version 6.0.1, and Version 6.0.2 processes form separate views if the Version 6.0 and Version 6.0.1 processes do not connect to Version 6.0.2 processes. To detect these views, examine the HMGR0218 messages that the various processes logged. If the processes connect, the same message is logged across all processes. Specifically, processes that are connected have the same view identifier and the same number of processes in the view.

  ```
  HMGR0218I: A new core group view is installed. The core group is
  defaultCoreGroup. The view identifier is (8:0.spoletoCell01\
  spoletoCellManager01\dmgr). The number of members in the new view is 2.
  ```

Setting up IP addresses for high availability manager communications

There are situations where you must select a preferred IP address, or a range of IP addresses that you want the high availability manager to use for communication within a core group.

Determine the transport protocol the core group associated with the high availability manager uses for communications.

If the core group is configured to use a point to point protocol for its communications, for example, channel framework or unicast, you should only set a preferred IP address if you want to restrict the high availability manager communications to a specific IP address.

If the core group is configured to use a multicast transport for its communications, you must set a preferred IP address and specify a range of multicast group IP addresses.

1. Configure a preferred IP address
   a. Make a list of all WebSphere processes on this machine. Include both application server processes and administrative processes, such as node agents or the deployment manager, in this list. If a preferred IP is set for one process on a machine, it should be set for all processes on that machine
   b. Determine the textual form of the preferred IP address. For example, 9.5.87.124 or 10.1.2.2.
   c. Update the IP address specified for the for the DCS unicast address for all processes identified in the first substep with the textual form of the preferred IP address.
      1) In the administrative console, for an application server process click **Servers > Application servers**, for a node agent process click **System administration > Node agents**, or for a deployment manager process click **System administration > Deployment manager**. Then select the appropriate process.
      2) Under **Additional properties** , click on **Ports** to bring up the list of ports for the selected process and then click on the port named **DCS_UNICAST_ADDRESS**.
      3) Enter the preferred IP address in the Host field and then click **Apply** to save your changes.

      Following are the allowed values for the Host field:

- * (an asterisk), which allows high availability manager communications on all NICs. This is the default value, but cannot be used if your processes must interact with V6 or V6.0.1 processes.
- Text IP address, such as 10.1.1.2, which restricts high availability manager communications to the specified NIC. Text IP addresses can be used if the host is configured to use either a single IP address or multiple IP addresses.
- A string host name of the form *myhost.mydomain* . A string host name cannot be used if the host is configured to use multiple IP addresses.

2. Specify a range of multicast group IP addresses for the high availability manager to use. If the core group transport is configured as multicast, you must associate a range of IP address with the NIC to be used for high availability manager communications. You must use core group properties to specify this range of IP addresses. Do not specify them on a per process basis.
   a. In the administrative console, click **Servers > Core groups > Core group settings**
   b. Select the desired core group.
   c. Specify the first IP address in the range associated with the NIC the high availability manager should use in the Multicast group IP start field
   d. Specify the last IP address in the range associated with that NIC in the Multicast group IP end field.
   e. Click **Apply** and then **Save** to save your changes.

The high availability manager uses the specified IP address or address range for communication within the core group.

Configuring the Discovery Protocol for a core group

The Discovery Protocol for a core group establishes network connectivity between a new core group member and the other members of that core group.

Understand the concepts that are described in “Core group Discovery Protocol” on page 357. Then determine how long you want the Discovery Protocol to wait before it recalculates the set of unconnected core group members and attempts to open connections to those members.

You might want to perform this task if you are trying to tune the discovery protocol behavior for a core group. The default value of 60 seconds, which is set during the WebSphere Application Server installation process, provides an acceptable process detection time for most situations.

To change wait interval for the Discovery Protocol:

1. In the administrative console, click **Servers > Core groups > Core group settings >** *core_group_name*.
2. Under Additional Properties, click **Custom Properties**.
3. Change the value that is specified for the IBM_CS_UNICAST_DISCOVERY_INTERVAL_SECS custom property.

   If the IBM_CS_UNICAST_DISCOVERY_INTERVAL_SECS property already exists, click on the property name and specify the new interval length, in seconds, in the Value field.

   If this property does not already exist, click **New** and create it:
   a. In the Name field, specify `IBM_CS_UNICAST_DISCOVERY_INTERVAL_SECS`.
   b. In the Value field specify the length of time, in seconds, you want the Discovery Protocol to wait before it recalculates the set of unconnected core group members and attempts to open connections to those members.
4. Click **OK** and then click**Review**.
5. Select **Synchronize changes with nodes**, and then click **Save**.
6. Restart all members of the core group.

After the servers restart, the core group members all run with the new Discovery Protocol setting.

Configuring the Failure Detection Protocol for a core group

The Failure Detection Protocol monitors the core group network connections that the Discovery Protocol establishes, and notifies the Discovery Protocol if a connection failure occurs.

- Understand the concepts that are described in “Core group Failure Detection Protocol” on page 358.
- Check your operating system settings that are relevant to TCP/IP socket closing events.
- Determine your failure detection goals and which settings must change to accomplish these goals.

You might want to perform this task if:
- You want to change the failover characteristics of your system.
- Your core groups are large and analysis indicates excessive CPU usage is spent monitoring heartbeats.

To change the settings for the Failure Detection Protocol:

1. In the administrative console, click **Servers > Core groups > Core group settings >** *core_group_name*.
2. Under Additional Properties, click **Custom Properties**.
3. Change the values specified for the IBM_CS_FD_PERIOD_SECS custom property. This property specifies the time interval, in seconds, between consecutive heartbeats. The default value for this property is 30 seconds.

   If the IBM_CS_FD_PERIOD_SECS property already exists, click on the property name, and in the Value field, specify the length of time, in seconds, that you want the Failure Detection Protocol to wait between consecutive heartbeats.

   If this property does not already exist, click **New** and create it:

   a. In the Name field, specify IBM_CS_FD_PERIOD_SECS.

   b. In the Value field specify the length of time, in seconds, that you want the Failure Detection Protocol to wait between consecutive heartbeats.
4. Change the values that are specified for the CS_FD_CONSECUTIVE_MISSED custom property. This property specifies the consecutive number of heartbeats that must be missed before the protocol assumes that the core group member has failed. The default value for this property is 6.

   If the CS_FD_CONSECUTIVE_MISSED properties already exists, click on the property name, and in the Value field, specify the number of heartbeats that must be missed before the Failure Detection Protocol assumes the core group member failed.

   If this property does not already exist, click **New** and create it:

   a. In the Name field, specify `CS_FD_CONSECUTIVE_MISSED`.

   b. In the Value field specify the number of heartbeats that must be missed before the Failure Detection Protocol assumes that the core group member failed.
5. Click **OK** and then click**Review**.
6. Select **Synchronize changes with nodes**, and then click **Save**.
7. Restart all of the members of the core group.

After the servers restart, the core group members all run with the new Failure Detection Protocol settings.

Configuring a core group for replication

WebSphere Application Server administrators can control the maximum amount of heap memory that the underlying core group transport can allocate. This memory is used for in-flight messages and network communication buffers. If you increase the maximum amount of heap memory that the transport can allocate, you must also increase the size of the transport buffer accordingly.

- Understand that other factors, such as the number of network interface cards on a machine, how the Network interface card is used, and network speed, can affect replication throughput performance.
- Determine the maximum amount of memory that you can let the core group transport allocate for buffering incoming messages. The default value is 10 megabytes. You can increase this value as required to allow for buffering of additional incoming messages. A setting of 100 is sufficient for most high throughput topologies.

You might want to perform this task if:

- You are trying to tune your systems replication performance.
- You are seeing large numbers of any of the following Distribution and Consistency Services (DCS) congestion messages in your SystemOut.log file:

```
DCSV1051W, a high severity congestion event for outgoing messages
DCSV1052W, a medium severity congestion event for outgoing messages
DCSV1054W, a medium severity congestion event for incoming messages
```

Important: Under extreme workloads, these messages might still occur on a properly tuned system.

To change amount of memory that is available for in-flight messages and network communication buffers:

1. Change the value of the IBM_CS_DATASTACK_MEG custom property. The value specified for the IBM_CS_DATASTACK_MEG custom property has a strong impact on the message throughput of a replication domain. The setting for this property controls the amount of memory that the data stack can use. The default value for this property is 50 megabytes. A replication domain that handles high throughput messaging, needs this property set to a higher value. The maximum value for this property is 256 megabytes. A setting of 100 megabytes is sufficient for most high throughput topologies.
   a. In the administrative console, click **Servers > Core groups > Core group settings >** *core_group_name*.
   b. Under Additional Properties, click **Custom Properties**.
   c. Change the value specified for the IBM_CS_DATASTACK_MEG custom property.

      If the IBM_CS_DATASTACK_MEG property already exists, click on the property name and, in the Value field, specify the maximum amount of memory that you want to let the core group transport allocate for buffering incoming messages.

      If this property does not already exist, click **New** and then:

      1) In the Name field, specify `IBM_CS_DATASTACK_MEG`.
      2) In the Value field, specify the maximum amount of memory you want to let the core group transport allocate for buffering incoming messages.

   d. Click **OK**, and then click **Save** to save your changes.
2. Change the size of the transport buffer. The maximum amount of heap memory that is specified for the IBM_CS_DATASTACK_MEG custom property should be less than or equal to the size of the transport buffer.
   a. In the administrative console, click **Servers > Application servers >** *server_name* **Core group service**.
   b. In the Transport buffer size field, specify, in megabytes the size of the transport buffer.
   c. Click **OK**, and then click **Save** to save your changes.
   d. Repeat this step for all of the core group members. Specify the same transport buffer size for all of the core group members.
3. Click **OK** and then click**Review**.
4. Select **Synchronize changes with nodes**, and then click **Save**.
5. Restart all members of the core group.

After the servers restart, they all run with the new replication settings.

Configuring core group IP caching

The caching of name-to-IP address information can be performed at many levels within the network communication software stack. These levels include within the operating system, the Java virtual machine, and within WebSphere Application Server components, such as core groups. Core groups use caching to reduce the overhead that is associated with IP address name lookup. You can adjust the interval at which a core group IP cache is cleared.

By default, a core group cache is cleared every 60 minutes. You can determine the correct time interval for your environment.

You want to change the length of time that IP addresses are retained in a core group cache.

To change how frequently a core group cache is cleared:

1. In the administrative console, click **Servers > Core groups > Core group settings >** *core_group_name*.
2. Under Additional Properties, click **Custom Properties**.
3. Change the value that is specified for the IBM_CS_IP_REFRESH_MINUTES custom property. A core group cache cannot be cleared more frequently then once a minute.

   If the IBM_CS_IP_REFRESH_MINUTES property already exists, click on the property name and in the Value field, specify the length of time, in minutes, you want to wait before a core group cache is cleared.

   If this property does not already exist, click **New** and create it:

   a. In the Name field, specify `IBM_CS_IP_REFRESH_MINUTES`.

   b. In the Value field specify the length of time, in minutes, that you want to wait before a core group cache is cleared.
4. Click **OK** and then click**Review**.
5. Select **Synchronize changes with nodes**, and then click **Save**.
6. Restart all members of the core group.

After the servers restart, all of the core group members run with the new discovery protocol setting.

Configuring core group socket buffers

Most operating systems provide program interfaces for performing operations involving the sending and receiving of data over sockets. Most operating systems also provide administrative capabilities to control the amount of memory allocated per socket that is used as data buffers.

- Check your operating system settings that are relevant to TCP sockets. For example, if you are using an AIX operating system, check the values that are specified for the tcp_sendspace, tcp_recvspace, and sb_max settings. Similarly on a Linux operating system, check the values that are specified for the tcp_rmem, and tcp_wmem settings.
- Use the WebSphere Application Server performance monitoring infrastructure to determine the average message size that the core group transport handles. If your operating system setting for the default buffer size is smaller than the average message size, make one of the following changes:
  - Change the default buffer size setting for your operating system. However, this action might be inappropriate because it might affect the operation of other applications running on this operating system.
  - Change the size of the socket buffer that the core group transport obtains. The value that is specified for the IBM_CS_SOCKET_BUFFER_SIZE core group custom property determines the size of the socket buffer that the core group transport obtains. The following table shows the relationship between the values that can be specified for this property and the underlying memory allocation size per socket buffer type:

Socket Buffer Type	Property set to No over rides	Property set to Small	Property set to Medium	Property set to Large
Unicast receive	Operating system default buffer size is used.	Buffer size is 64 kilobytes	Buffer size is 256 kilobytes	Buffer size is 1 megabyte
Unicast send	Operating system default buffer size is used.	Operating system default buffer size is used.	Buffer size is 64 kilobytes	Buffer size is 128 kilobytes
Multicast receive	Operating system default buffer size is used.	Buffer size is 512 kilobytes	Buffer size is 1 megabyte	Buffer size is 3 megabytes

You might want to change the size of your core group buffers in the following circumstances:

- You are directed to do so by IBM Support
- You are directed to do so during the course of installing another WebSphere product.
- You want to change the behavior of the core group transport without affecting the behavior of other sockets.
- You are trying to tune the network communication path of your system to your application.

To change the socket buffer space that the core group transport allocates:

1. In the administrative console, click **Servers > Core groups > Core group settings >** *core_group_name*.
2. Under Additional Properties, click **Custom Properties**.
3. Change the value that is specified for the IBM_CS_SOCKET_BUFFER_SIZE custom property.

   If the IBM_CS_SOCKET_BUFFER_SIZE property already exists, click the property name and specify either `No over rides`, `small`, `medium`, or `large`.

   If this property does not already exist, click **New** and create it:

   a. In the Name field, specify `IBM_CS_SOCKET_BUFFER_SIZE`.

   b. In the Value field specify one of the following strings:
      - `No over rides`
      - `small`
      - `medium`
      - `large`
4. Click **OK** and then click**Review**.
5. Select **Synchronize changes with nodes**, and then click **Save**.
6. Restart all members of the core group.

After the servers restart, the core group members all run with the new socket buffer size settings.

Specifying a core group when adding a node

By default, a cell contains a single core group. In this situation, during node federation, a node agent is created and automatically added to this core group. However, if the cell contains multiple core groups, you must specify the core group to which the node agent is assigned.

If the cell to which you are adding a node agent contains multiple core groups, determine the core group to which you want this node agent to belong.

You have a cell that contains multiple core groups and you want to select the core group to which a newly created node agent is added.

To add a newly created node agent to a specific core group:

Include the coregroupname parameter on the **addNode** command. For example:

```
addNode dmgr_host dmgr_port  -coregroupname existing_core_group_name
```

The node agent resides in the specified core group.

Specifying a core group when creating an application server

By default, a cell contains a single core group. In this situation, whenever you add an application server to that cell, it is automatically added to this core group. However if the cell contains multiple core groups, you must specify the core group to which you want the application server assigned.

If the cell to which you are adding an application server contains multiple core groups, determine the core group to which you want this application server to belong.

You have a cell that contains multiple core groups and you want to select the core group to which a newly created application server is added.

To add a newly created node agent to a specific core group:

1. In the administrative console, click **Servers > Application servers > New** .
2. Select the node on which you want to create the application server.
3. Specify a name for the new application server, and then click **Next**.
4. Select the server template that you want for this application server, and then click **Next**. Usually, you want to use the template for the default server.
5. Select the core group to which you want this application server to belong from the list of available core groups, and then click **Next**.
6. Confirm the selections on the summary presented, and click **Finish**.
7. Click **OK**, and then click **Save** to save your changes.
8. Select **Synchronize changes with nodes** and click **Save** again.
9. Restart all members of the core group.
10. Start the server.

After the server starts, it becomes a member of the selected core group.

Viewing the core groups in a cell

A core group is a component of the high availability manager. A default core group, called DefaultCoreGroup, is created for each cell. A core group can contain application servers, proxy servers, node agents, and the deployment manager. A core group must contain at least one node agent or the deployment manager.

If you need to move servers between core groups, in preparation for the move, you can view the list of the different core groups contained in a cell to help determine to which core group you want to move a server.

To view the core groups that are in a cell:

In the administrative console, click **Servers > Core groups > Core group settings**.

A list of the core groups that are in the cell is displayed.

Click the name of one of the core groups to obtain specific information about that core group.

Core group collection

A core group is a component of the high availability manager function. A default core group, called DefaultCoreGroup, is created for each cell in the WebSphere Application Server environment. A core group can contain standalone servers, cluster members, node agents and the deployment manager. A core group must contain at least one node agent or the deployment manager.

To view this administrative console page, click **Servers > Core Groups > Core group settings** .

Click **New** to define a new core group. After a core group is defined, several fields become read-only. To change those fields, delete and redefine the core group.

Click **Delete** to delete a core group. A core group must be empty before it can be deleted.

Name

Specifies the name of the core group. Click the core group name to edit the settings for that core group. This field is read-only.

Description

Specifies a description of the core group. This field is read-only.

Connected core groups

Specifies the core groups that are connected to this core group by access points. This field is read-only.

Viewing core group members

A core group member is an application server, proxy server, the deployment manager, or a node agent that is a member of a high availability core group.

If you need to move servers between core groups, in preparation for the move, you can view the list of servers that belong to each core group to help determine which servers you want to move to a different core group.

To view the members of a core group:

1. In the administrative console, click **Servers > Core groups > Core group settings**. A list of core groups that are in the cell is displayed.
2. Click the name of one of the core groups and then click **Core group servers**.

A list of the core group members is displayed.

In the administrative console, you can click **Servers > Core groups > Core group settings > Preferred coordinator servers** to determine which of the core group members are on the list of preferred coordinator servers for this core group. You should remove any servers that you intend to move to a different core group from this list before changing their core group association.

Core group servers collection

Use this page to view the servers that are part of a core group. A core group server can be an application server, a deployment manager, or a node agent that is a member of a high availability core group. Use this page to move servers into a different core group. All members of a cluster must be in the same core group. If you select one or more members of a cluster, all of the members of that cluster must be moved.

To view this administrative console page, click **Servers > Core groups > Core group settings > New** or select an existing core group for editing. Then click **Core group servers**.

Name	Specifies the names of the servers in the core group. This field is read-only. Click this field to specify custom properties for this server.
Node	Specifies the node that contains the core group server. This field is read-only.
Version	Specifies the product version of the node in the cell. A Version 6 deployment manager node can own a Version 5 managed node. Version 5 managed nodes are easily identified in this field. This field is read-only.
Type	Specifies the server process type, which can be either deployment manager, node agent, or application server. Standalone application servers in the cell, managed nodes, and cluster members all display as application server types. This field is read-only.
Cluster Name	Specifies the cluster name if the core group server is part of a cluster. If the core group server does not belong to a cluster, this field is blank. This field is read-only.

To move one or more servers to another core group, select the check box next to the names of the servers that you want to move and click **Move**.

Important: You must stop a core group server before you move it to another core group.

Core group server settings

Use this page to create or change custom properties for a server in a core group.

To view this administrative console page, click **Servers > Core groups > Core group settings > *Select an existing core group* > Core group servers > *Select an existing server***.

Extended information about the fields on the panel:

Node	Specifies the name of the node that contains the core group server. This field is read-only.
Name	Specifies the name of the core group server. This field is read-only.
Custom Properties	Click on this field to define or edit a custom property for the core group server.

Creating a new core group

A default core group, called DefaultCoreGroup, is created for each cell. This default core group is sufficient in most configurations. However there are some circumstances under which you need to create additional core groups for a cell.

Determine how to segment your existing cell into multiple core groups. Use the following rules as guidelines:

- All members of a cluster must be in the same core group. A core group can contain multiple clusters.
- Core group members cannot cross firewall boundaries.
- Put clusters with direct relationships in the same core group. Examples of direct relationships include:
  - A single application is deployed on multiple clusters.
  - An application on one cluster calls an application on another cluster.
- Each core group must contain at least one deployment manager or node agent process.

You might want to add another core group to a cell under the following circumstances:

- A firewall separates members of an existing core group, for example a proxy server and an application server are separated by a firewall.
- You are scaling up the number of servers in a cell. Multiple core groups are recommended for a large cell.

To create a new core group:

1. In the administrative console, click **Servers > Core groups > Core group settings > New** .
2. In the Name field, specify a unique name for the new core group. The name can contain alpha and numeric characters, but not the following special characters:

   ```
   # \ / , : ; " * ? < > | = + & % '
   ```

   The name also cannot begin with a period (.) or a blank space. A blank space does not generate an error. However, leading and trailing blank spaces are automatically deleted from the name.
3. Add a description of this core group that helps other administrators understand the purpose of this core group.
4. Click **OK** and then click**Review**.
5. Select **Synchronize changes with nodes**, and then click **Save**.

The cell contains another core group.

You must now:

- Complete your core group configuration. The initial core group settings and policies are derived from a template. If the settings from the default template do not meet your requirements, you can:
  - Change the number of coordinators for this core group.
  - Change the transport type for this core group.
  - Add policies for this core group.
- Move members to the new core group.
- Create bridges between core groups. If clusters with direct relationships are not in the same core group, set up a core group bridge to connect the related core groups.

.

Moving core group members

When moving members to a different core group, remember that: each WebSphere process is a member of exactly one core group, all members of a given cluster must belong to the same core group, and each core group must contain at least one deployment manager or node agent process.

- Review core group concepts.
- Determine which core group members you want to move, and to which core group you want to move them
- Stop affected processes using the following guidelines:
  - Case 1: You are not moving the deployment manager. Stop all of the processes you are moving.
  - Case 2: You are moving the deployment manager. Because the deployment manager has to be moved separately, stop all of the other processes you are moving, but leave the deployment manager running until after you have moved all of these other processes.

    Important: In general, you should not move a deployment manager.

You might need to move one or more core group members:

- To populate a newly created core group.
- To rebalance existing core groups.

To move members between core groups:

1. In the administrative console, click **Servers > Core groups > Core group settings >** *core_group_name* **> Core group servers**.
2. Select the core group containing the processes that you want to move to another core group.
3. In the Select column, select the servers that you want to move. If you are populating a new core group, at least one node agent or deployment manager must be moved to the new core group.
4. Click **Move**. The **Core groups > DefaultCoreGroup > Core group servers > Move** administrative console panel is displayed showing the servers you want to move and the core group to which these servers currently belong.
5. Indicate in the To core group field the core group to which you want these servers moved.
6. Click **Apply** and then **Save**.
7. Click **System administration > Nodes**, and then click **Synchronize** to synchronize your changes on all running nodes.
8. Manually synchronize all stopped nodes by running the **syncNode** command from the *profile_root*/*node_agent_profile*/bin directory.
9. If the deployment manager is moved, restart the deployment manager process.
10. Restart all of the other moved processes.

After the servers complete their restart, all moved servers should belong to their new core group.

- You can verify that the servers are in the correct core groups. For each core group, in the administrative console, click **Servers > Core groups > Core group settings >** *core_group_name* **> Core group servers** and look at the list of core group servers that displays.
- You can set up core group bridges if any of the core groups need to communicate with each other. See "Core group communications using the core group bridge service" on page 404 for more information.

Core group server move options

Use this page to move one or more core group servers to a different core group. You must stop a core group server before you move it.

Servers can be moved from one core group to another, as long as the following core group requirements are not violated:

- A non-empty core group retains at least one node agent or deployment manager as a member of that group. (The high availability manager configuration change listeners are only available on the node agent or deployment manager servers.
- All members of a cluster must be members of the same core group. If one or more of the servers you are moving belongs to a cluster, you must move all of the members of that cluster. (A core group can span multiple WebSphere clusters.)

To view this administrative console page, click **Servers > Core groups > Core group settings >** *core_group* **> Core group servers**. Select the servers to be moved and then click **Move**.

Extended information about the core group fields:

Move selected servers	Specifies the servers that you selected to be moved. It cannot be edited.
From core group	Specifies the name of the core group that you are moving the servers from. It can not be edited.
To core group	Specifies the core group to which these servers will belong.

Click **Apply** to make your changes effective. Click **Save** and save and synchronize your changes with all managed nodes.

Disabling or enabling a high availability manager

A unique HAManagerService configuration object exists for every core group member. The enable attribute in this configuration object determines if the high availability manager is enabled or disabled for the corresponding process. When the enable attribute is set to `true`, the high availability manager is enabled. When the enable attribute is set to `false`, the high availability manager is disabled. By default, the high availability manager is enabled. If the setting for the enable attribute is changed, the corresponding process must be restarted before the change goes into effect. You must use the wsadmin tool to disable or enable a high availability manager.

- Determine if you need to use a high availability manager to manage members of a core group.
- Add a script file that is similar to the following file to your system. This sample script file disables the high availability manager on a specific process. You can modify this script file if you need to disable the high availability manager on all of the processes in a cell or on all of the processes that are members of a specific core group. You can also modify this script file to enable a high availability manager that you previously disabled.

```
#####################################################################
# Script name = disableHamOnProcess.pty
#####################################################################

def getHAMServiceOnAll():
    # get a list of all HAManagerService objects in the cell.
    processes = AdminConfig.list("HAManagerService").split("\n")
    rc = []
    for p in processes:
        p = p.strip()
        rc.append(p)
    return rc

# The HAManagerService ObjectName has the following format
#cells/cellname/nodes/nodename/servers/servname:hamanagerservice.xml
#id def getNodeName(service):
    # The 4th /-separated element in the service name is the node name
    n = service.split("/")[3]
    return n

def getProcessName(service):
    # The 6th /-separated element in the service name is the process name
    p = service.split("/")[5]
    return p.split("|")[0]

def printHelp():
    print "This script disables the HA Manager on a specific process"
    print "Format is disableHamOnProcess nodeName processName"

################################
# main
################################
if(len(sys.argv) > 1):
    # get node name and process name from the command line
    nodeName = sys.argv[0]
    processName = sys.argv[1]
    # get a list of all HAManagerService objects in the cell.
    processes = getHAMServiceOnAll()
    for p in processes:
        # debug
        print "Checking process "+p
        # Check for a node name match.
        n = getNodeName(p)
        if (nodeName == n):
            # node name matches, check for server name match
            pn = getProcessName(p)
            if (pn == processName):
                # both node and process names match. Found the one we
```

```
                # are looking for. Disable and exit.
                print "Disabling the HA Manager on process ",
                print p
                AdminConfig.modify(p, [["enable", "false"]])
                AdminConfig.save()
                break
else:
    printHelp()
```

You might want to disable a high availability manager if you are trying to reduce the amount of resources, such as CPU and memory, that WebSphere Application Server uses and have determined that the high availability manager is not required on some or all of the processes in a core group.

You might need to enable a high availability manager that you previously disabled because you are installing applications on core group members that must be highly available.

You must use the wsadmin tool to disable a high availability manager or to enable a high availability manager that you previously disabled.

1. Start the wsadmin tool.
2. Start the deployment manager for the cell that contains the high availability manager that you are disabling or enabling. The deployment manager for this cell must be running for the sample script to work.
3. Use the wsadmin tool to issue the following command to start the script:

   `-lang jython -f` *`script_file_name node_name process_name`*

   For example, to start the sample script for the WASDOCLNodePH02 node and the ClusterMember2 process, issue the following command:

   ```
   -lang jython -f disableHamOnProcess.pty WASDOCLNodePH02 ClusterMember2
   ```
4. Restart all processes with changed setting for the enable attribute.

The processes start with the high availability manager in the appropriate state.

Viewing high availability group information

High availability groups are dynamically created components of a core group. They cannot be configured directly, but they are directly affected by static data, such as policy configurations, which is specified at the core group level. You can use the administrative console to view information about the high availability groups that are part of a core group.

You need to know the name of the core group that contains the high availability groups you want to view. You should also determine if you need to view all of the high availability groups or a subset of these groups, based on the group name.

You might want to perform this task if:

- You want to view the current set of high availability groups.
- You want to view the group name of a high availability group.
- You want to view the policy that is associated with a high availability group.
- You want to view the state of a high availability group

To view information about the high availability groups contained in a core group:

1. In the administrative console, click **Servers > Core groups > Core group settings**.
2. Click the core group that contains the high availability groups you want to view.
3. Click the **Runtime** tab.
4. Specify a value in the **Group name** field.

- Specify an asterisk in this field if you want a list of all the high availability groups that are part of this core group.
- Specify a set of name-value pairs, separated by a comma, to get a list of only those high availability groups that contain the specified name-value pairs in their group name. For example, you might specify the following value to obtain a list of all of the high availability groups that contain `IBM_hc=MyCluster` and `type=WAS_TRANSACTIONS` in their names.

  ```
  IBM_hc=MyCluster,type=WAS_TRANSACTIONS
  ```

5. Click **Show groups**.

A list of high availability groups that are contained in this core group that meet the specified criteria is displayed, with pertinent information about these groups.

You can click on the name of one of the high availability group names to display information about members of that group. Because high availability group members are dynamically created, no member information is available for configured servers that are not actually running.

You can also view the distribution of active high availability group members.

Viewing the distribution of active high availability group members

Because high availability group members are dynamically created, core group policy configuration is used to determine which high availability group members the high availability manager activates. In some situations, it is possible for active members of multiple high availability groups to all be running on the same server process. You can use the administrative console to view the distribution of active high availability group members across your application servers.

You need to know the name of the core group that contains the high availability groups you want to view. You should also determine if you need to view all of the high availability groups or a subset of these groups, based on the group name.

You might want to perform this task if:

- You want to view the current active high availability group member distribution for the servers in a core group.
- You want to determine whether a particular server is overloaded because it is hosting the active member for multiple high availability groups.
- You want to check the current active member distribution before you update policies that might affect this distribution.
- You want to verify that you obtained the proper results to the current active member distribution after a policy change goes into affect.

To view the distribution of active high availability group members:

1. In the administrative console, click **Servers > Core groups > Core group settings**.
2. Click the core group that contains the high availability groups you want to view.
3. Click the **Runtime** tab.
4. Specify a value in the **Group name** field.
   - Specify an asterisk in this field if you want a list of all the high availability groups that are part of this core group.
   - Specify a set of name-value pairs, separated by a comma, to get a list of only those high availability groups that contain the specified name-value pairs in their group name. For example, you might specify the following value to obtain a list of all of the high availability groups that contain `IBM_hc=MyCluster` and `type=WAS_TRANSACTIONS` in their names.

     ```
     IBM_hc=MyCluster,type=WAS_TRANSACTIONS
     ```

5. Click **Show servers**.

A list of servers displays that shows the number of active high availability group members on each server.

Servers with active members collection

Use this page to determine how many high availability group members are active on a particular application server.

To view this administrative console page, click **Servers > Core groups > Core group settings >** *core_group*. Click on the **Runtime** tab and specify group name properties for a high availability group. (You can specify an asterisk (*) to get a list of the servers that are hosting active members for all the high availability groups in this core group.) Then select **Show severs**.

Server

Specifies the name of a server on which there are active high availability group members. This field is read-only.

Node

Specifies the node on which each server is running. This field is read-only.

Version

Specifies the version of the WebSphere Application product on which each node is running. This field is read-only.

Active members

Specifies the number of high availability group members that are currently active on that server. This field is read-only.

High availability groups collection

Use this page to view information about the high availability groups contained in a core group.

To view this administrative console page, click **Servers > Core groups > Core group settings >** *core_group*. Click on the **Runtime** tab. In the Group name properties field, specify a match criterion for a specific high availability group, or specify an asterisk to get a list of all the high availability groups that are part of this core group. (A match criterion is one or more name=value pairs that match attributes contained in a high availability group's name.) Then click **Show groups**.

High availability group
Specifies the names of the high availability groups. The name of a high availability group is a set of name-value pairs or attributes, separated with commas. For example, `name=productiongroup,policy=abc,ibm=websphere` could be the name of a high availability group. This field is read-only.

Quorum
Specifies if quorum is enabled for each high availability group. This field is read-only.

Policy

Specifies the policies that have match criteria that matches properties contained in the name of that high availability group. There should only be one policy listed for a high availability group. However, if multiple policies have match criteria that equally match properties in a high availability group's name, all of the policies with matching criteria are listed, and the ERROR icon appears in the status column.

For example, if you have a high availability group named `name=productiongroup,policy=abc,ibm=websphere`, and MyPolicy1 has the match criteria `name=productiongroup`, and MyPolicy2 has the match criteria `policy=abc`, both MyPolicy1 and MyPolicy2 are considered matching policies and are listed in the Policy column.

This field is read-only.

Status

Specifies, with icons, whether or not only one policy is associated with a high availability group. If the OK icon displays in this column, a single policy is associated with that high availability group. If the ERROR icon displays in this column, multiple policies are associated with that group.

If the ERROR icon displays for a high availability group, you must adjust the match criteria for one or more of the policies listed in the Policy column for that group so that the correct policy is the only one associated with that high availability group.

The match criteria for multiple policies can match some of the same properties in a group's name as long as one policy has a match criteria that matches more of the properties in that group's name than the match criteria of any of the other policies. For example, if you have a high availability group with a name that consists of the following name and value pairs:

`name=productiongroup,policy=abc,ibm=websphere`

and MyPolicy1 has the match criteria `name=productiongroup` and MyPolicy2 has the match criteria `name=productiongroup,ibm=websphere`, MyPolicy2 is considered the matching policy because it has more match criteria that matched the properties contained in the name of the high availability group.

This field is read-only.

Use the **Disable** button to disable all of the members of a high availability group that were previously active or idle. One of the few times you might want to use this button is if you are planning to remove or delete all of the servers on which this group has a member running.

Use the **Enable** button to enable all of the members of a high availability group that were previously disabled. These members can then be activated according to the policy associated with that group.

High availability group members collection

Use this page to view information about the individual members of a high availability group. This page lists the current members of the selected high availability group.

To view this administrative console page, click **Servers > Core groups > Core group settings >** *core_group*. Click on the **Runtime** tab. In the Group name properties field, specify a match criterion for a specific high availability group, or specify an asterisk (*) to get a list of all the high availability groups that are part of this core group. (A match criterion is one or more name-value pairs that match attributes contained in the name of a high availability group.) Then click **Show groups** and select one of the high availability groups listed.

Name
Specifies the name of a high availability group member.

Node
Specifies the node on which each high availability group member is running.

Version
Specifies the version of the WebSphere Application Server product on which each node is running.

Status
Specifies the state of the high availability group members.

High availability group members are either idle, activated, or disabled. The usual states are idle or activated. One of the few times you might want to disable a member is if it is running on a server that you plan to remove or delete.

- If a group member is idle, it cannot be assigned any work.
- If a group member is activated, it can be assigned work.
- If a group member is disabled, it must be enabled before it can be activated.

Click the **Disable** button to prevent a group member from participation in the group. A member in the disabled state can never be made active or used by the group.

Click the **Enable** button to enable a group member that was previously disabled.

Click the **Activate** button to activate an idle group member.

Click the **Deactivate** button to make an active group member idle.

Creating a policy for a high availability group

Every high availability group has to have an associated policy. This policy determines which members of a high availability group to put in the active state.

Before creating a new policy, you should review the following topics:

-
- “High availability group policy modification guidelines” on page 367
- “High availability group policies” on page 361

You should also know:

- The name of the core group that you want to associate with the new policy.
- The name of the high availability group that you want this policy to control.
- The function, such as transaction log recovery or messaging engine, that is associated with this high availability group.
- The policy types, such as One of N or Static, that this function supports.
- The type of policy you want to create.
- The policy settings, such as failback, and preferred servers only, that you want to configure for this policy.

WebSphere Application Server includes default policies that are already associated with the high availability groups some of the WebSphere Application Server components use. If these default policies do not meet the requirements of your installation, it is recommended that you create a new policy instead of changing one of the default policies. The creation of new policies provides you with the capability to tailor the policy settings to your installations requirements while giving you the option to revert back to the default policy.

To create a new policy:

1. In the administrative console, click **Servers** > **Core groups** > **Core group settings** *core_group_name* > **Policies** > **New**.
2. Select the new policy you want in effect for a specific high availability group. If you need to define a new policy, the policy options are:
   - All active policy: All of the group members are activated.
   - M of N policy: *M* group members are activated. The number that is represented by *M* is defined as part of the policy details.
   - No operation policy: No group members are activated.
   - One of N policy: Only one group member is activated.
   - Static policy: The active members of a group are statically configured.
3. Click **Next**.
4. Specify a name for the policy in the Name field. The name must be unique within the scope of the core group. Make the name meaningful to other administrators.
5. **Optional:** Specify a Description of the policy in the description field. This description might include the name of the associated core group.
6. Specify a value for the Is alive timer field, if the default value is too long or too short a time period. This value determines how frequently the high availability manager checks the health of the high availability group members. The default value is 0 seconds.
   - If you specify -1 (minus 1), the Is alive timer is disabled.
   - If you specify 0 (zero), the value that is specified for the Is alive timer at the core group services level is used for high availability groups that are associated with this policy.
   - If you specify an integer between 1 and 2147483647, inclusive, this value is used for the high availability groups that are associated with this policy.
7. Make sure the Quorum field is not selected. You should not enable Quorum unless you are explicitly instructed to do so in the documentation for some other product.
8. Select the **Failback** field if you want to have the high availability manager make the most preferred member the currently active member whenever this action is possible. This option is available only for M of N and One of N policies.
9. Select the **Preferred servers only** field if you want the high availability manager to only activate group members on servers that are contained in the Preferred servers list. This option is available only for M of N and One of N policies. If you select this option, you must configure a list of preferred servers. A description of how to set up this list is provided in a later optional step.
10. Specify the number of group members that you want active in the Number of active members field. This option is available only for an M of N policy.
11. Click **Apply** and then select **Match criteria**.
12. On the next panel, click **New** and then configure the match criterion for this policy.
    a. In the Name field, specify the name of one of the name-value pairs contained in the name of the high availability group that you want to associate with this policy.
    b. In the Value field, specify the value of the name-value pair you specified in the Name field.
    c. **Optional:** In the Description field, add a description of this match criterion . For example, you might specify First attribute to indicate that this name-value pair matches the first attribute contained in the group name.

d. Click **OK**.

e. Repeat these steps for each additional attribute you want to include as part of your match criterion.

You should set the match criterion for a new policy to two or more of the name-value attributes that are contained in the name of the high availability group to ensure that this policy is used instead of one of the WebSphere Application Server default policies. Using this example, the following high availability group-to-policy association is established:

13. Uunder **Additional Properties**, select the **Static group servers** field to configure the list of servers that you want activated. This option is available only for Static policies. Click **Add** to move core group servers into the list of Static group servers, and then Click **OK** after you complete the list.
14. **Optional:** Under Additional Properties, select **Preferred servers** and select the preferred servers for this policy. This option is available only if you selected the Preferred servers only field for M of N and One of N policies. If you do not set up this list, no group members are activated.

    Click **Add** to move core group servers into the list of preferred servers.

    Select specific servers in the list and click **Move up** and **Move down** to adjust the order of the servers within the list. Make sure that the most preferred server is at the beginning of the list and the least preferred server is at the end of the list.

    After you complete the preferred servers list, click **OK**.

    Important: Use caution when selecting preferred servers. WebSphere Application Sever cannot detect if you select an inappropriate server as a preferred server. For example, if the policy affects a messaging engine or transaction service, only select preferred servers from the messaging engine cluster. Similarly, if the policy affects a transaction service, only select preferred servers from the transaction service cluster.
15. Click **OK** and then click**Review**.
16. Select **Synchronize changes with nodes**, and then click **Save**.

The new policy goes into affect after it is saved and synchronized. You do not have to stop and restart the affected application servers.

You can change the **Failback** and **Preferred servers only** options for this policy without stopping and restarting the affected application servers.

You can create or update the list of preferred servers that for this policy without stopping and restarting the affected application servers.

Core group policies

Use this page to create or update the various high availability group policies. For a given high availability group, the associated policy determines which members of the group should be made active.

To view this administrative console page, click **Servers > Core groups > Core group settings > New** or *existing core group* **> Policies**.

Click **New** to define a new policy. After a policy is defined there are several fields that you can no longer change. To change those fields, delete and redefine the policy. Click **Delete** after selecting a policy to delete the selected policy.

After adding a high availability group, you need to take more actions to enable workload balancing for messaging resources. For more information about the extra actions, see the Related tasks.

All of the policy fields on this page are read-only. To change the values specified in any of these fields, click on the name of the policy you want to change. When the console page **Core group settings** > *group_name* > **Policies** > *policy name* displays, you can edit the policy properties.

Name
Specifies the name of the policy.

Description
Specifies a description of the policy.

Policy type
Specifies the desired policy type.

Restrictions:

1. If you are setting up a policy for a transaction manager, you must select One of N as the policy type.
2. If you are setting up a policy for a service integration bus you must select One of N or Static as the policy type. The default policy that IBM provides for a service integration bus uses a One of N policy type.

Following is a list of valid policy types:

All active
The All active policy indicates that the high availability manager keeps all of the application components that are running on all of the servers in the high availability group active at all times.

M of N
The M of N policy is similar to the One of N policy. However, it enables you to specify the number (M) of high availability group members that you want to keep active if it is possible to do so. The number of active members must be greater than one and less than or equal to the number of servers in the high availability group. If the number of active servers is set to one, this policy is a match for the One of N policy.

No operation
The No operation policy indicates that no high availability group members are made active.

One of N
The One of N policy keeps one member of the high availability group active at all times. This is used by groups that desire singleton failover. If a failure occurs, the high availability manager starts the singleton on another server.

Static The Static policy allows you to statically define or configure the active members of the high availability group.

Match criteria
Specifies one or more name-value pairs that are used to associate this policy with a high availability group. These pairs must match attributes that are contained in the name of a high availability group before this policy is associated with that group.

Core group policy settings

Use this page to define a policy for a high availability group. A policy is defined at the core group level. It only applies to matching high availability groups contained within this core group

To view this administrative console page, click **Servers > Core groups > Core group settings > New** or *existing core group* > **Policies** > *policy_name*.

Name

Specifies the name of the policy. This name must be unique within the scope of a core group.

Policy type

Specifies the policy type that was selected when this policy was created. This is a read-only field. If you want to change the policy type, you must delete this policy and then create it again specifying a different policy type. If this is an IBM provided policy, do not delete it. Instead create a new policy and specify more of the attributes contained in the name of the high availability group as the match criterion for this new policy. The policy with the greatest number of matches to attributes in a group's name is the policy that is associated with that group.

Description

Specifies a description of this policy. For example, the clustered TM policy provided with the product has ″TM One-Of-N Policy″ as its description.

Is alive timer

Specifies, in seconds, the interval of time at which the high availability manager will check the health of the active group members that are governed by this policy. If a group member has failed, the server on which the group member resides is restarted.

The high availability manager detects two fundamentally different kinds of failures.

- An entire process failure. This failure detection is accomplished using functions such as the heartbeat timers. This type of detection does not involve the Is alive timer function. If an entire process fails, it will be detected and the various high availability groups will fail over to other servers in the core group.
- An application or program failure. If, for some reason. an application or a program, like the transaction manager or a service integration bus function hangs, the high availability manager will eventually detect the hang. The amount of time that might pass before the hang is detected is determined by the value specified for the Is alive timer parameter. The parameter controls how often the high availability manager will call back to the component that created the high availability group member and ask if it is still alive. This allows detection of hung code or program errors that somehow do not cause the entire process to stop functioning or terminate.

Data type	Integer • Valid values are -1 to 600 seconds, inclusive. • If -1 (negative 1) is specified, this function is disabled. • If 0 (zero) is specified, the frequency at which the high availability manager checks the health of the active group members is determined by the time interval specified at the application server process level. • If a value larger than 0 (zero) is specified, the high availability manager uses the time interval specified here, instead of the one specified at the application server process level, when determining how frequently it should check the health of the high availability group members using this policy.
Default	0 (zero)

Quorum

Specifies whether quorum checking is enabled for a group governed by this policy. Quorum is a mechanism that can be used to protect resources that are shared across members of the group in the event of a failure.

Important: Quorum is an advanced hardware function and should not be enabled unless you thoroughly understand how to properly use this function. If not used properly, this function can cause data corruption.

The Quorum setting in the policy will only have an effect if the following items are true:

- The group members are also cluster members.
- GroupName.WAS_CLUSTER=*clustername* must be specified as a property in the group name of any high availability group matching this policy.

When enabled, any group using this policy will not achieve quorum until a majority of the members are running. For example, if there are n members in the group, (n/2) + 1 servers must be online in order to achieve quorum. No group members will be activated until quorum has been achieved.

The quorum mechanism is designed to work in conjunction with a hardware control facility that allows application servers to be shut down if a failure causes the group to be partitioned.

Fail back
Specifies whether work items assigned to the failing server are moved to the server that is designated as the most preferred server for the group if a failure occurs. This field only applies for M of N and One of N policies.

Preferred servers only
Specifies whether group members are only activated on servers that are on the list of preferred servers for this group. This field only applies for M of N and One of N policies.

Number of active members
Specify how many of the high availability group members are to be activated. This field only applies for the M of N policy.

Additional Properties
Specifies one or more of the following options, depending on the type of policy you selected:

Custom properties	Click to specify custom properties for the policy.
Match criteria	Click to set up a match criterion for the policy.
Preferred servers	Click to set up a list of servers that are given preference when group members are activated.
Static group servers	Click to set up a list of the specific servers that are activated.

New core group policy definition
Use this page to create a new policy for a high availability group.

When you create a new policy, the first page that displays lets you select a policy type. To view the administrative console page where you select a policy type, click **Servers > Core groups > Core group settings > New** or select an existing core group. Then click **Policies > New**.

Select one of the following policy types:
- All active policy: Under this policy, all of the group members are activated.
- M of N policy: Under this policy, *M* group members are activated. The number represented by *M* is defined as part of the policy details.
- No operation policy: Under this policy, no group members are activated.
- One of N policy: Under this policy, only one group member is activated.
- Static policy: Under this policy, only specified group members are activated.

After selecting a policy, click **Next** to continue.

Preferred servers

Use this page to define the ordered list of *preferred servers* for the selected policy. The policy gives preference to the servers in this list when activating group members.

To view this administrative console page, you must be working with a policy that has a policy type of M of N or One of N. If your policy has one of these policy types, click **Servers > Core groups > Core group settings > New >** or *select an existing core group* **> Policies > New >** or *select an existing policy*. Under **Additional Properties**, select **Preferred Servers**.

Use **Add** and **Remove** to move servers into and out of the list of preferred servers. Use **Move up** and **Move down** to adjust the order within the list of preferred servers. Make sure that the most preferred server is at the top of the list and the least preferred server is at the bottom.

Click **OK** to make your changes effective. Click **Save** to save and synchronize your changes with all managed nodes.

Changes to the preferred servers list take affect as soon as they are saved and synchronized. You do not have to stop and restart the affected application servers.

Match criteria collection

Use this page to view the match criteria that are defined for a policy.

To view this administrative console page, click **Servers > Core groups > Core group settings > New >** or *select an existing core group* **> Policies > New** or *select an existing policy* **> Match criteria**.

Click **New** to create a new match criterion for the policy. Click the name of a match criterion to change any of that criterion's properties.

Name

Specifies the name portion of a name-value pair that is part of the name of the high availability group that you are associating with this policy.

Value

Specifies the value portion of a name-value pair that is part of the name of the high availability group that you are associating with this policy.

Description

Specifies a description of the match criterion. Make the description meaningful. For example, the description might indicate the high availability group that this name-value pair matches.

Match criteria settings

Use this page to define a match criterion for a policy.

To view this administrative console page, click **Servers > Core groups > Core group settings > New >** or *select an existing core group* **> Policies > New** or *select an existing policy* **> Match Criteria >** *criterion name*.

The name and value fields should match a name-value attribute included in the name of a high availability group you want associated with this policy.

After you define a match criterion, click **Apply** to make your changes effective. Click **Save** to save and synchronize your changes with all managed nodes.

Name
Specifies the name portion of a name-value pair that is part of the name of the high availability group that you are associating with this policy.

Value
Specifies the value portion of a name-value pair that is part of the name of the high availability group that you are associating with this policy.

Description
Specifies a description of the match criterion. Make the description meaningful. For example, the description might indicate the high availability group that this name-value pair matches.

Static group servers collection
Use this page to designate for a static policy which high availability group members should be made active.

This option is available under Additional Properties only if Static is selected as the policy type. To view this administrative console page, click **Servers > Core groups > Core group settings >** *existing_core_group* **> Policies >** *static_policy_name* > **Static group servers**.

Use **Add** and **Remove** to move servers into and out of the list of servers that should be activated. Only high availability group members that are associated with this policy appear on this page.

After you finish updating the list, click **Apply** to make your changes effective. Click **Save** to save and synchronize your changes with all managed nodes.

Selecting the policy for a high availability group
Every high availability group has an associated policy. The high availability manager uses this policy to determine which members of a high availability group to put in the active state.

Before you select a policy for a high availability group, you should review the following topics:
-
- “High availability group policy modification guidelines” on page 367
- “High availability group policies” on page 361

You should also know:
- The name of the core group that you want to associate with the new policy.
- The name of the high availability group that you want this policy to control.
- The function, such as transaction log recovery or messaging engine, that is associated with this high availability group.
- The policy types, such as One of N or Static, that this function supports.
- The type of policy you want to create.
- The policy settings, such as failback, and preferred servers only, that you want to configure for this policy.

You have multiple policies defined for a high availability group, and you want to specify which of these policies the high availability manager uses to govern the group.

To select a policy for a high availability group:
1. In the administrative console, click **Servers > Core groups > Core group settings** *core_group_name* .

2. Click the **Runtime** tab to determine both the name of the high availability group, and the name of the policy that is currently controlling the group. See “Viewing high availability group information” on page 391 for more information on how to perform this step. You must have at least one of the group members running.
3. Click the **Configuration** tab to determine the match criteria defined in the current high availability group policy.
4. Use the information you obtained in the previous steps and update the match criteria for the policy you are selecting. The match criterion must contain all of the match criteria from the original policy, and at least one additional attribute from the name of the high availability group.
5. Click **OK** and then click**Review**.
6. Select **Synchronize changes with nodes**, and then click **Save**.

The high availability manager uses the new policy to govern the designated high availability groups.

Specifying a preferred server for messaging requests

If a core group includes a cluster of application servers, and a messaging engine is configured for that cluster, any of the servers in that cluster can handle work items for the messaging engine. The default message provider in WebSphere Application Server is based on Service Integration Bus (SIB) technology, and is governed by the Default SIBus policy, which is a One of N policy. This policy ensures that only one of the application servers in the cluster is active at a time). You can modify the high availability group policy to specify that a specific cluster member handles the messaging work.

Before specifying a preferred server for messaging requests:

- You should review the following topics:
  - “High availability groups” on page 360
  - “High availability group policy modification guidelines” on page 367
  - “High availability group policies” on page 361
- You must determine:
  - The name of the core group that includes the server that you want to handle messaging requests.
  - The name of the high availability group for the messaging function.
  - The name of the policy that is associated with this high availability group
- You must create a new policy specific to the high availability group that controls the messaging engine cluster, if one does not already exist.

  It is possible for a single policy to govern several different high availability groups. Therefore, to modify the policy for cluster scoped control, you must create a new policy specific to the high availability group that controls the messaging engine cluster. See “Creating a policy for a high availability group” on page 395 for more information on how to create this policy.

  After you create the new policy and associate the policy with the high availability group for a given cluster, You can specify a preferred server for messaging requestws..

For high availability, you must configure a messaging engine to run in a cluster. However, you might want a specific cluster member to handle the messaging requests. Another member of the cluster should handle the messaging requests only if the preferred member fails.

1. In the administrative console, click **Servers > Core groups > Core group settings >** *core_group_name*.
2. Click the **Runtime** tab to determine both the name of the high availability group, and the name of the policy that is currently controlling the group. See “Viewing high availability group information” on page 391 for more information on how to perform this step. You must have at least one of the group members running.

3. In the administrative console, click **Servers** > **Core groups** > **Core group settings** > *core_group_name* > **Policies**.
4. Click the name of the policy that you want to modify.
5. Under Additional Properties, select **Preferred servers** and select the preferred servers for this policy.

   Click **Add** to move core group servers into the list of preferred servers.

   Select specific servers in the list and click **Move up** and **Move down** to adjust the order of the servers within the list. Make sure that the most preferred server is at the beginning of the list and the least preferred server is at the end of the list.
6. After you complete the preferred servers list, click **OK**.
7. Click **OK** and then click**Review**.
8. Select **Synchronize changes with nodes**, and then click **Save**.

All work items for the messaging engine on the associated cluster are routed to the new preferred server.

Configuring the core group bridge service

The core group bridge service can be configured for communication between core groups. A core group is a statically defined component of the high availability manager. To configure communication between core groups, use an access point group. An access point group is a collection of core groups that communicate with each other.

Review the following topics before you configure a core group bridge service:

- "Core groups" on page 349, which describes a core group.
- "Creating a new core group" on page 387, which describes how to configure a core group.
- "Configuring communication between core groups that are in the same cell" on page 420, which describes how to configuring communication between core groups that are in the same cell.

You must configure the core group bridge service whenever two or more core groups are configured in the same cell. You can also configure the core group bridge to share traffic among core groups that are in different cells. Configure the core group bridge to communicate between cells only when the service is required by another WebSphere Application Server component. By configuring the core group bridge service, the availability status of the servers in each core group is shared among all the configured core groups. For more information, see "Core group communications using the core group bridge service." You can configure core groups to communicate in the following ways:

- Use the core group bridge for communication between core groups that are in different cells. Configuring this type of communication is the most common core group scenario. You can configure each cell to communicate with one or more other cells. For more information, see "Configuring the core group bridge between core groups that are in different cells" on page 406.
- Use advanced configurations. You might need to configure core group communication between core groups that are in the same cell, that communicate across different networks, or that use a proxy peer. For more information, see "Creating advanced core group bridge configurations" on page 414.

Multiple core groups can communicate with each other.

Continue configuring the high availability environment. See Chapter 9, "Setting up a high availability environment," on page 345 for more information.

Core group communications using the core group bridge service

The core group bridge service can be configured to share availability information about internal WebSphere Application Server components between core groups. For example, by configuring the core group bridge service, each core group can be aware of the status of all of the application servers that are

configured in all of the core groups. Use access point groups to define the core groups that communicate. Do not use the core group bridge service to share application information among core groups.

A core group is a statically defined component of the high availability manager. Each cell must have at least one core group. WebSphere Application Server creates a default core group called **DefaultCoreGroup** for each cell. For more information about core groups, see “Core groups” on page 349. Two or more core groups can be set up to communicate with each other and share workload management information by defining access point groups. The core groups that communicate can be in the same cell or in different cells.

Core group bridge overview

To configure communication between core groups, you must configure an *access point group*. An access point group is a collection of the core groups that communicate with each other. Add a *core group access point* to the access point group for each core group that needs to communicate.

A *core group access point* is a collection of server, node, and transport channel chain combinations that communicate for the core group. Each core group has one or more defined core group access points. The **DefaultCoreGroup** has one default core group access point. However, you might consider configuring more than one core group access point for a core group if that particular core group needs to be connected to other core groups that are on different networks. See “Advanced core group bridge configurations” on page 415 for more information about configuring core groups to communicate across different networks.

The node, server, and transport channel chain combinations that are in a core group access point are called *bridge interfaces*. A server that hosts the bridge interfaces is a *core group bridge server*. The transport channel chain defines the set of channels that are used to communicate with other core group bridge servers. Each transport channel chain has a configured port that the core group bridge server uses to listen for messages from other core group bridge servers.

Each core group access point must have at least one core group bridge server. The core group bridge server provides the bridge interface for each core group access point. Because core group bridge servers within a core group access point serve as backups for each other, it is recommended that you have two core group bridge servers within each core group access point. Then, if one core group bridge server fails, the other core group bridge server can take over the failed core group bridge server’s responsibilities.

If you are configuring communication between core groups that are in the same cell, create one access point group and add a core group access point for each core group that needs to communicate. See “Advanced core group bridge configurations” on page 415 for more information about configuring communication between core groups that are in the same cell.

If you are configuring the core group bridge between core groups that are in different cells, you still use an access point group. However, you must create and configure the access point group for each cell. Each cell has an access point group that contains a core group access point for the core group that is in the cell, and a *peer access point* for each peer cell.

A peer access point references a core group access point that is configured in a different cell. Each access point group must have one peer access point for each different cell. Do not configure multiple peer access points that reference the same cell.

Each peer access point has one or more *peer ports* or one *proxy peer access point*.

A peer port corresponds to a bridge interface that is defined in the peer cell. You can define several peer ports for each peer access point.

Define a proxy peer access point if the peer access point cannot be reached directly by using a peer port, but can be reached by using another peer access point. The proxy peer access point specifies a peer access point that can communicate with the peer core group that cannot be reached directly. The proxy peer must have defined peer ports. Specify one proxy peer or one or more peer ports, but not both. See “Advanced core group bridge configurations” on page 415 for more information about proxy peer access points.

The following diagram shows a core group bridge configuration between two different cells that is using peer access points with peer ports.

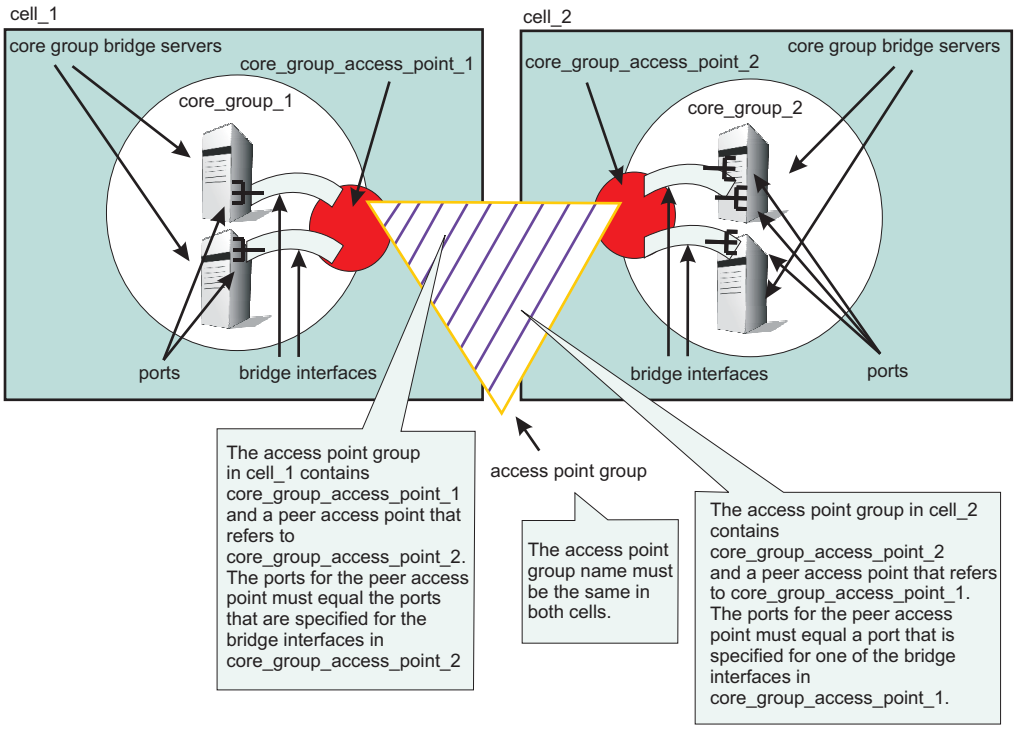


Figure 1. Core group bridge configuration in two different cells

Configuring the core group bridge between core groups that are in different cells

The core group bridge service can be configured for communication between core groups. Use an access point group to define the core groups that communicate. Use this task to configure communication between core groups that are in different cells.

Make sure that:

- You have two or more core groups that are in different cells. Core groups are a statically defined component of the high availability manager. See “Core groups” on page 349 for more information about core groups.
- Any cell that uses core group bridges to connect to core groups in other cells have a name that is unique when compared to the names of the other cells. See “Configuring communication between core groups that are in the same cell” on page 420 for more information.
- Any core group that uses a core group bridge is configured with Channel framework as its transport mechanism.

Use the core group bridge service to share the availability status of the servers in each core group among all the configured core groups. Enable the core group bridge only when the service is required by a WebSphere Application Server component.

For example, the core group bridge is required when configuring an Enterprise JavaBeans (EJB) backup cluster. Configure a backup EJB cluster in a different cell to continue servicing requests when the primary cluster fails. See “Creating backup clusters” on page 322 for more information.

When you configure core group communication between core groups that are in different cells, you must configure an access point group to connect the core groups. The name of the access point group must be the same in all of the connected cells. This task uses the DefaultAccessPointGroup access point group, but you can create and use another access point group.

You can define the CGB_ENABLE_602_FEATURES custom property on all of the access point groups in your configuration if you want to be able to add core group bridge servers to the configuration without restarting the other servers in the configuration. After you enable this property, you can add a core group bridge server in one cell, without modifying the configuration in the other cell to include peer ports for the core group bridge server. Instead of manually configuring peer ports in each of the cells, you can configure the peer ports in one cell and let the other cell discover the peer ports for the first cell.

If you decide to use the CGB_ENABLE_602_FEATURES custom property, you must decide which cell is the listener cell and which cell initiates contact with the other cells before you begin your configuration. The listener cell does not need to contain any peer access points or peer ports for the other cells in the configuration. For example, in a configuration that contains a secured cell and an unsecured cell, configure the unsecured cell as the listener. The unsecured cell cannot access information about the secured cell. Configure the core group bridge service on the listener cell first.

Attention: Do not configure the CGB_ENABLE_602_FEATURES custom property if you already configured the FW_PASSIVE_MEMBER custom property. Any server for which the configured the FW_PASSIVE_MEMBER custom property is configured cannot initiate contact with other systems in the configuration. For more information about the FW_PASSIVE_MEMBER custom property, see “Core group bridge custom properties” on page 426.

Complete the following set of steps for each of the cells in your configuration.

1. Configure bridge interfaces for your core group access point. Configuring a bridge interface indicates that the specified node, server, and chain combination is a core group bridge server. This node and server use the specified chain to communicate with other core groups.
   a. In the administrative console, click **Servers > Core groups > Core group bridge settings > Access point groups >** *DefaultAccessPointGroup* **> Core group access points >** *CGAP_1\DefaultCoreGroup* **> Show detail > Bridge interfaces > New**.
   b. Select a node, server, and transport channel chain that becomes your bridge interface.
   c. Click **Apply**.
   d. Repeat this set of steps to add more bridge interfaces to the core group access point. Define at least two bridge interfaces for each core group access point to back up your configuration. By defining two bridge interfaces, you define two core group bridge servers. If one core group bridge server fails, the other can take over the work so that the communication between the core groups can continue.

      Important:
      The bridge interfaces that you select must all have the same transport channel chain.
2. If you want the ability to add core group bridge servers to the configuration without restarting the other servers in the configuration, define the CGB_ENABLE_602_FEATURES custom property on all of the access point groups in your configuration.

a. In the administrative console, click **Servers > Core groups > Core group bridge settings > Access point groups >** *DefaultAccessPointGroup* **> Custom properties > New**.

b. Type the name as CGB_ENABLE_602_FEATURES and set the value to any string.

The existence of the CGB_ENABLE_602_FEATURES property enables the property. Therefore, you can set the value to any string value. Setting the value to false does not disable the property. To disable the property, you must remove it from the list of defined custom properties or change its name.

c. Click **Apply** and save your configuration.

When you complete this step on all the access point groups in your configuration, you can add a bridge interface to one of the cells. You can save the configuration so that it is propagated to all of the nodes. Instead of restarting all of the application servers, you need to restart the new bridge interface server only.

3. Add peer access points and peer ports to your access point group.

If you defined the CGB_ENABLE_602_FEATURES custom property for all of the access point groups in your configuration, you do not need to add peer access points or peer ports to the listener cell.

Add a peer access point for each core group that is in another cell. Within each peer access point, you should configure a peer port that corresponds to each bridge interface in the other cell. Before you add a peer access point, you should have the following information about the other cell:

- cell name
- core group name
- core group access point name
- host and port information. The host and port correspond to the bridge interfaces that are configured in the other cell. Specify a peer port for each bridge interface that is in the other cell.

a. In the administrative console, click **Servers > Core groups > Core group bridge settings > Access point groups >** *DefaultAccessPointGroup* **> Peer access points > New**.

b. Specify the information for your peer access point and click **Next**.

c. Select **Use peer ports**. Specify the host and port information for your peer cell. For example, if you defined a bridge interface in *cell_x*, use that configuration information for your peer port in *cell_y*.

d. Click **Next** and then **Finish**. Save your configuration.

If more than one bridge interface is defined in your peer cell, add additional peer ports for each bridge interface.

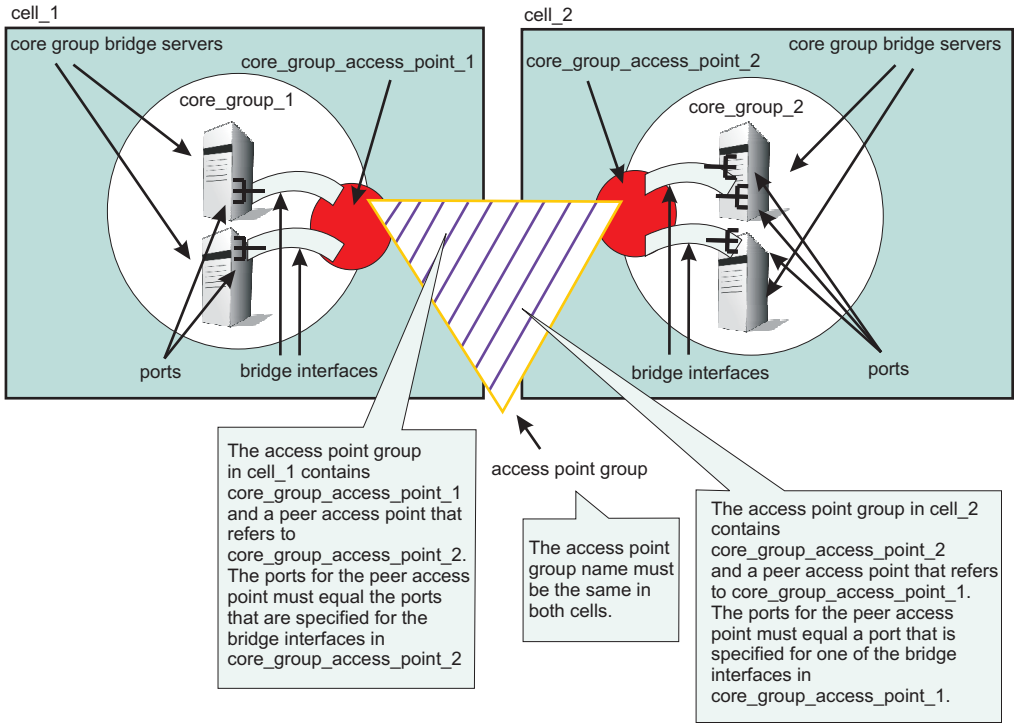
a. Click **Peer_access_point_name > Show detail > Peer ports > New**.

b. Enter the host name and port.

c. Click **Apply** and save your changes.

You configured the core group bridge between core groups that are in different cells.

The following illustration is an example of a configuration between two core groups that are in two different cells. Each cell has a defined *DefaultAccessPointGroup* access point group, which contains one core group access point for the core group that is in the cell and a peer access point for the other cell.



Continue configuring the high availability environment. See Chapter 9, “Setting up a high availability environment,” on page 345 for more information.

Core group bridge settings

The core group bridge is the service that enables communication between core groups. A core group is a statically defined component of the high availability manager. Use this page to view the structure of your access point groups. Access point groups link core groups that are in the same cell or in different cells and allow the core groups to communicate with each other.

To view this administrative console page, click **Servers > Core groups > Core group bridge settings**.

Each access point group is a collection of core group access points. If you are configuring communication between core groups that are in the same cell, configure an access point group with a core group access point for each core group in the cell. If you are configuring communication between core groups in different cells, configure an access point group that has one core group access point for the local cell and a peer access point for each other cell.

Each core group access point has one or more bridge interfaces. Each peer access point has a proxy peer or one or more peer ports. A bridge interface is a server that is configured to communicate with other core groups by using a particular transport channel chain. Click **Access point groups** to configure the settings for each access point group that is configured.

- **Access point group** - An access point group defines the core groups that communicate with each other. Each access point group consists of a collection of core groups.
  - **Core group** - Specifies a core group that is in this access point group. Core groups are referenced by core group access points.
    - **Core group access point** - The core group access point defines the set of servers that provide access to the core group. Bridge interfaces define the servers that are in the core group access point.

- **Bridge interface** - A bridge interface defines a node, server, and chain for the core group access point. The chain defines the transport channels that the server uses for receiving information. Typically, all the bridge interfaces in a core group access point use the same chain.

  Important: The bridge interfaces listed in the display tree on this page are listed as `*,port`. The asterisk symbol represents the multi-home support for DCS instead of a specific hostname.

– **Peer core group** - Specifies a core group in a different cell. Define peer access points to communicate with peer core groups.
  - **Peer access point** - Each peer access point is used to communicate a core group in a different cell. Each peer access point corresponds to a core group access point that is in the peer cell. Use one or more peer ports or one proxy peer to define the communication settings.
    - **Peer port** - Each peer port identifies a bridge interface of a core group bridge in the peer cell.
    - **Proxy peer** - A proxy peer is used to identify the communication settings for a peer access point that cannot be accessed directly through peer ports. A proxy peer specifies a peer access point that can communicate with the destination core group. The specified proxy peer must be a peer access point that has defined ports.

Access point group collection

Use this page to view the sets of access point groups. Access point groups define the set of core groups that communicate with each other. Access point groups that connect multiple cells must have one core group access point and a single peer access point for each remote cell. Access point groups that provide communications between core groups in the same cell must contain only core group access points.

To view this administrative console page, click **Servers > Core groups > Core group bridge settings > Access point groups**.

Name:

Specifies the name of the access point group. The access point group name must be unique within the cell.

Access point group settings:

Use this page to modify the core group access points and the peer access points that belong to this access point group. An access point group defines the set of core groups that communicate with each other. Group the access points to support communication. Access points can be either peer access points or core group access points. Define core group access points so that core groups in the same cell can communicate. Define peer access points so that core groups in different cells can communicate.

To view this administrative console page, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name*.

From this page, you can edit the core group access points or the peer access points that belong to the access point group.

Name:

Specifies the name of the access point group. The access point name must be unique within a cell.

Core group access point collection

Use this page to configure your set of core group access points. Core group access points define the set of servers that provide access to the core group. At least one core group access point must be defined for each core group in the local cell.

To access this administrative console page, click **Servers** > **Core groups** > **Core group bridge settings** > **Access point groups** > *access_point_group_name* > **Core group access points**.

Available core group access points:

A list of core group access points that are available to add to the access point group.

Core group access points in** access point group**:

A list of core group access points that are in the specified *access point group*.

Core group access point settings:

Use this page to configure your core group access points. The core group access point defines the set of core group bridge servers that provide access to the core group. A core group bridge server is an application server that is configured to run the core group bridge service. Define unique core group access points for each different network over which you want the core group in this cell to connect to other core groups that are defined in another cells. The core group access point has a collection of bridge interfaces. Each server that is used as a bridge must have a unique bridge interface for every core group access point in the core group.

For example if you define access points AccessPointA and AccessPointB and ServerX and ServerY are configured as core group bridges in a core group, ServerX must have unique bridge interfaces for both AccessPointA and AccessPointB. The ServerY server must also have unique bridge interfaces for both AccessPointA and AccessPointB.

When a you create a core group, a core group access point is automatically created. Do not delete the last access point in a core group.

The core group access point that is automatically defined belongs to a default access point group. You can use the default core group access point and access point group to configure communication between core groups. You must create and configure bridge interfaces for the default core group access point. See “Bridge interface settings” on page 412 for more information about bridge interfaces.

To access this administrative console page, click **Servers** > **Core groups** > **Core group bridge settings** > **Access point groups** > *access_point_group_name* > **Core group access points** >*core_group_access_point_name* > **Show detail**.

Name:

Specifies the name for the core group access point.

Core group:

Specifies the core group that is associated with this core group access point.

Bridge interface collection

Each core group access point has a collection of bridge interfaces. This collection defines the interfaces on the set of servers that provide access to the core group. All the servers in this collection have the core group bridge service enabled. The core group bridge service provides communication between core groups. A bridge interface defines the node, the server, and the chain for the core group access point. The chain defines the transport channels that are used by the server for receiving information.

To access this administrative console page, click **Servers** > **Core groups** > **Core group bridge settings** > **Access point groups** > *access_point_group_name* > **Core group access points** >*access_point_name* > **Show detail** > **Bridge interfaces**.

Server:

Specifies the node and the server combinations that are bridge interfaces for the core group access point.

Transport channel chain:

Specifies the transport channel chain that is used for transport by the bridge interface. For all the core group access points in an access point group, the transport channel chains must resolve to the same host. To ensure that the transport channel chains resolve to the same host, use the same chain for all of the core group access points in an access point group.

Bridge interface settings:

Use this page to specify the bridge interfaces that provide access to the core group access point. A bridge interface is a particular node and server that runs the core group bridge service. The core group bridge service is the service that provides communication between core groups.

A bridge interface is defined by a unique combination of a node, server, and transport chain. You cannot configure a cluster of servers to run the same bridge interface. A transport chain represents a network protocol stack that is operating within an application server.

To access this administrative console page, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Core group access points >** *access_point_access_point_name* **> Show detail > Bridge interfaces >** *server_node*.

Bridge interfaces: Select one of the listed server, node, and chain combinations that are available to become bridge interfaces for your core group access point.

Bridge interface creation:

A bridge interface specifies a particular node and server that runs the core group bridge service. A bridge interface is defined by a unique combination of a node, a server, and a transport chain. A transport chain represents a network protocol stack that is operating within an application server.

To access this administrative console page, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Core group access points >***access_point_name* **> Show detail > Bridge interfaces > New**.

Available bridge interfaces:

Specifies the node, server and transport channel chain combinations that are available to become bridge interfaces for this core group access point. Only bridge interfaces with transport chains that contain Transmission Control Protocol (TCP) inbound channels using the same port name as existing bridge interfaces in this core group access point are displayed. The bridge interfaces that are already used by any core group access point are not displayed.

Peer access point collection

Use this page to view a list of peer access points. Peer access points are used to communicate with core groups that are in other cells. A peer access point collection is the set of peer access points that are used to communicate with the core group access point in this access point group. Specify a single peer access point for each remote cell. This collection cannot contain two peer access points for the same cell.

To view this administrative console page, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Peer access points**.

Available peer access points:

Specifies a list of peer access points that are available to join the access point group.

Peer access points in** access_point_group**: Specifies a list of peer access points that are in the selected access point group.

Peer access point settings:

Use this page to configure a peer access point. Each peer access point is used to communicate with core groups in other cells. A peer access point corresponds to a core group access point in the peer cell. The peer access point communication settings are specified by using one or more peer end points or a proxy peer.

A peer access point must contain either peer ports or a proxy peer access point, but not both. When the peer access point is directly accessible within its access point group, specify peer ports. When the peer access point can be reached only indirectly, use a proxy peer access point. A proxy peer access point is used to identify the communication settings for the peer access point that cannot be accessed directly. The proxy peer access point specifies a peer access point that can communicate with the appropriate destination core group. The specified proxy peer access point must be a peer access point that has defined ports.

To view this administrative console page, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Peer access points >** *peer_access_point_name* **> Show detail**.

Name:

Specifies the name of the peer access point. The name must be unique within the local cell.

Cell:

Specifies the cell in which the peer access point resides.

Core group:

Specifies the core group in which the peer access point resides.

Core group access point:

Specifies the name of the core group access point that is in the peer cell.

default	defaultCoreGroupAccessPoint

Use peer ports:

Specifies that you are using peer ports instead of a proxy peer access point. Use peer ports when the peer access point is directly accessible within its access point group. Click **Peer ports** to specify the peer ports for the peer access point.

Use proxy peer access point:

Specifies that you are using a proxy peer access point instead of peer ports. A proxy peer is defined when the peer access point can be reached only indirectly through another peer access point. A proxy peer is used to identify the communication settings for the peer access point that cannot be accessed directly. The proxy peer specifies a peer access point that can communicate with the destination core group. The specified proxy peer must be a peer access point that has defined peer ports.

Proxy peer access point:

Specifies the specific peer access point that is used to access a core group.

Peer port collection

Use this page to define the peer ports for the peer access point. Each peer port identifies a bridge interface of a core group bridge service in the peer cell. Each peer access point that does not have a proxy peer must have one or more peer ports.

To view this administrative console page, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Peer access points >** *peer_access_point_name* **> Show detail > Peer ports**.

Host:

Specifies the host name that is used by the bridge interface in the remote cell.

Port:

Specifies the port that is used by the bridge interface in the remote cell.

Peer port settings:

Use this page to configure a peer port. A peer port identifies the host name and port of an application server that is a bridge interface in another cell. This application server is using the core group bridge service to communicate with other core groups. Each peer access point can have one or more peer ports. Each port identifies a bridge interface of a core group bridge service in the peer cell.

To view this administrative console page, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Peer access points >** *peer_access_point_name* **> Show detail > Peer ports >** *peer_port_name*.

Host:

Specifies the host name on which the core group bridge in the remote cell is listening.

Port:

Specifies the port number that is associated with the host on which the core group bridge in the remote cell is listening.

Creating advanced core group bridge configurations

Use this task to configure core groups to communicate with each other.

Configure two or more core groups that need to communicate with each other. For more information about core groups, see “Core groups” on page 349. To configure core groups, see “Creating a new core group” on page 387.

If you are configuring core group communication between core groups that are in different cells, see “Configuring the core group bridge service” on page 404. Communication between core groups that are in different cells is the most common usage scenario. Using this task, you can configure core group communication between core groups that are in the same cell or that use a proxy peer to communicate across 3 cells.

- Configure core group communication between core groups that are in the same cell. Configuring communication between any core groups that are in the same cell is required. For more information, see “Configuring communication between core groups that are in the same cell” on page 420.

- Configure communication between core groups using a proxy peer access point. Sometimes, your core group might not have access to the core group that you want to communicate with. However, if you can access a core group that can communicate with the inaccessible core group, you can create a proxy peer access point. For more information, see “Configuring core group communication using a proxy peer access point” on page 423.

Multiple core groups can communicate with each other.

Continue configuring the high availability environment. See Chapter 9, “Setting up a high availability environment,” on page 345 for more information.

Advanced core group bridge configurations

This topic describes advanced core group bridge configurations. These configurations are not performed as often as the typical core group bridge between core groups that are in different cells.

Advanced configuration scenarios

The most common core group bridge configuration is between two core groups that are in different cells on a single network. See “Core group communications using the core group bridge service” on page 404 for more information about this common scenario. The scenarios that are described in this topic are for advanced configuration situations.

There are four types of communication between core groups that you can configure:

- Communication between core groups that are in the same cell
- Communication within the cell and outside of the cell
- Communication between core groups across different networks
- Communication between core groups using a proxy peer access point

Communication between core groups that are in the same cell

All core groups that are in the same cell must be configured to communicate with each other. To configure core group communication within a cell, create one access point group with one core group access point for each core group. Select one or more servers to be core group bridge servers, and define a bridge interface for each server. All the bridge interfaces that are in an access point group that connects core groups that are in the same cell must have a node, server, and chain combination that resolves to the same port. To make sure all the bridge interfaces resolve to the same port, you can configure all the bridge interfaces use the same chain name. The following image shows an example of three core groups that are in the same cell and are connected by one access point group. The sample configuration shows how communication between core groups in the same cell is configured in the administrative console.

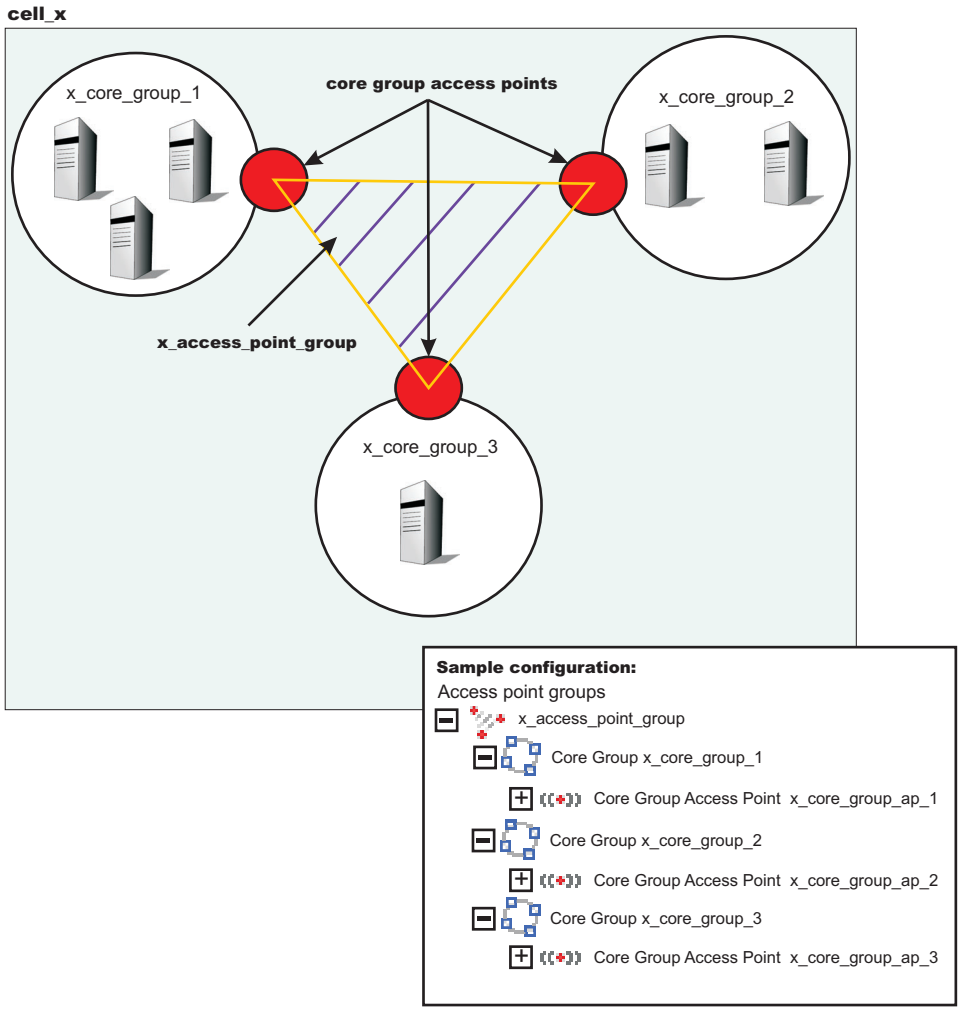


Figure 2. Communication between core groups that are in the same cell

See “Configuring communication between core groups that are in the same cell” on page 420 for more information.

Communication within the cell and outside of the cell

The following example illustrates a configuration between three core groups that are in three different cells. Each cell has one access point group for communication between core groups in the cell. Each cell also has a defined *access_point_group_xyz* access point group, which contains one core group access point group for the core group that is in the cell, and one core group access point for each of the core groups in the other two cells.

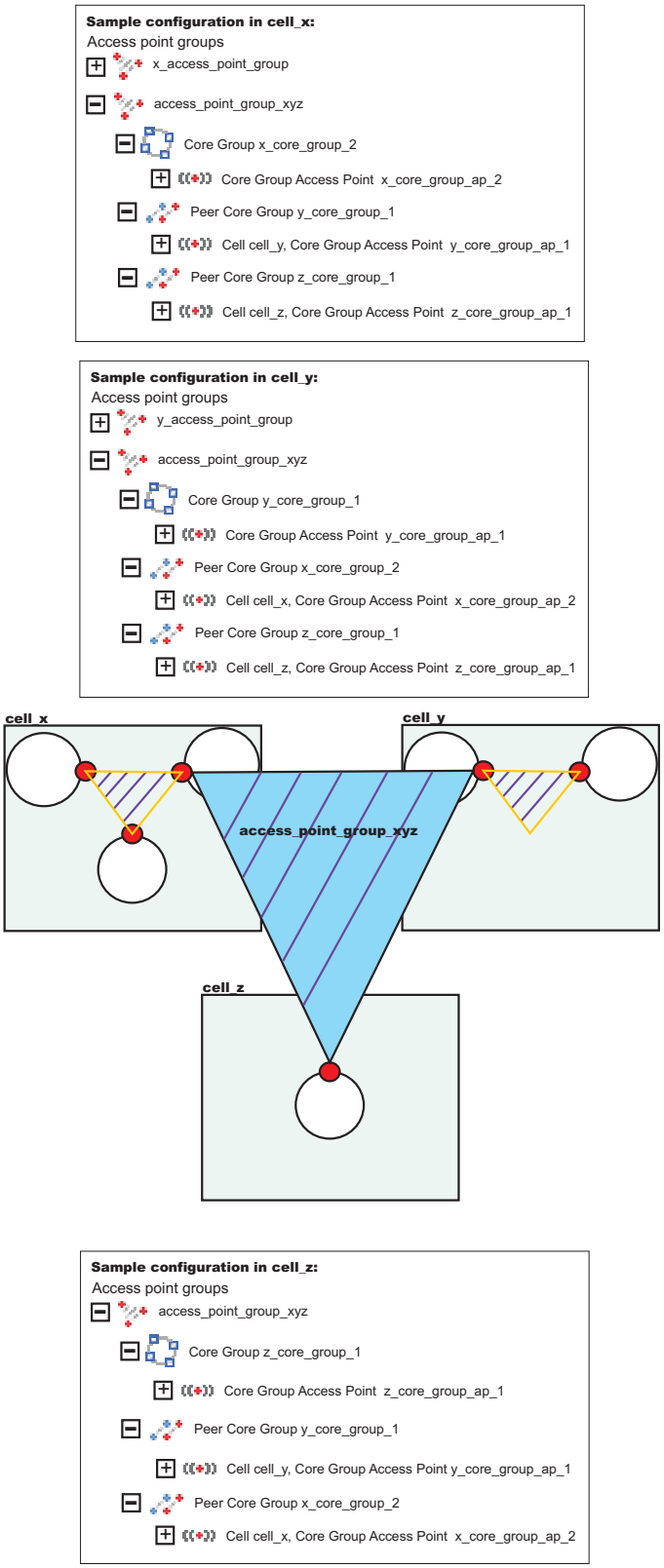


Figure 3. Communication between core groups that are in the same cell with core groups outside of the cell

The following example shows the relationship between bridge interfaces and peer ports for the communication between the*cell_x* cell and the *cell_z* cell. In the *cell_x* cell, two bridge interfaces are

defined. In the *cell_z* cell a peer access point exists for the *x_core_group_ap_2* core group access point with peer ports defined that correspond to the bridge interface information that is defined in the *cell_x* cell .

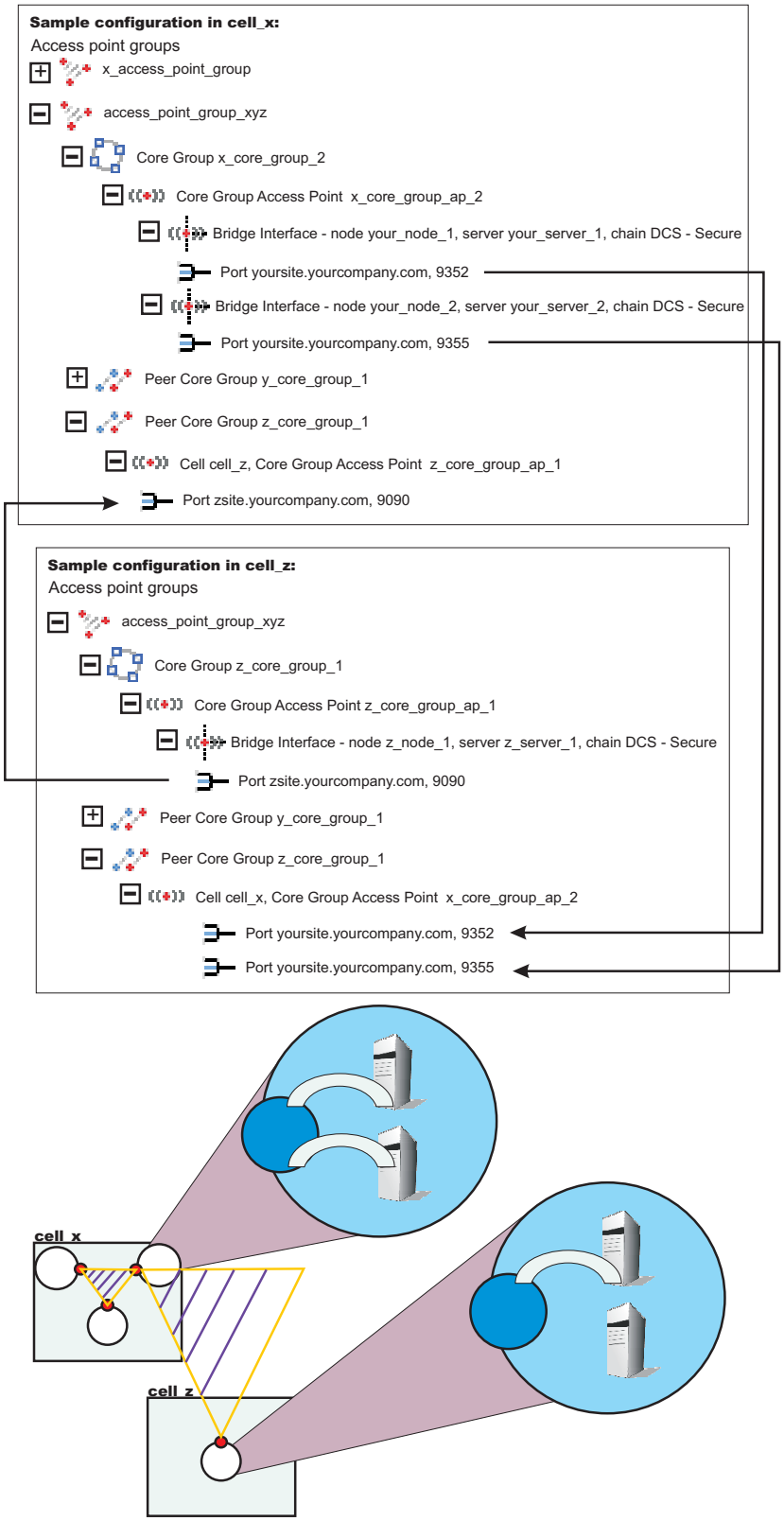


Figure 4. Bridge interfaces in one cell correspond to peer ports in the other cell

As a result, the *core_group_x* , *core_group_y* and *core_group_z* core groups can communicate with each other.

Communication between core groups across different networks

In this scenario, one core group is configured to communicate with two or more core groups in different cells across two or more networks. For example, a core group in the *cell_x* cell needs to communicate with core groups in the *cell_y* and *cell_z* cells. Create two access point groups in the *cell_x* cell. The*access_point_group_xy* access point group, in the *cell_x* cell contains a core group access point and a peer access point for the core group in the *cell_y* cell. The *access_point_group_xz* access point group in the *cell_x* cell contains a core group access point and a peer access point for the core group in the *cell_z* cell. The *cell_y* cell has an *access_point_group_xy* access point group, which has a core group access point and a peer access point for the *cell_x* cell. The *cell_z* cell has an *access_point_group_xz*access point group, which has a core group access point and a peer access point for the *cell_x* cell.

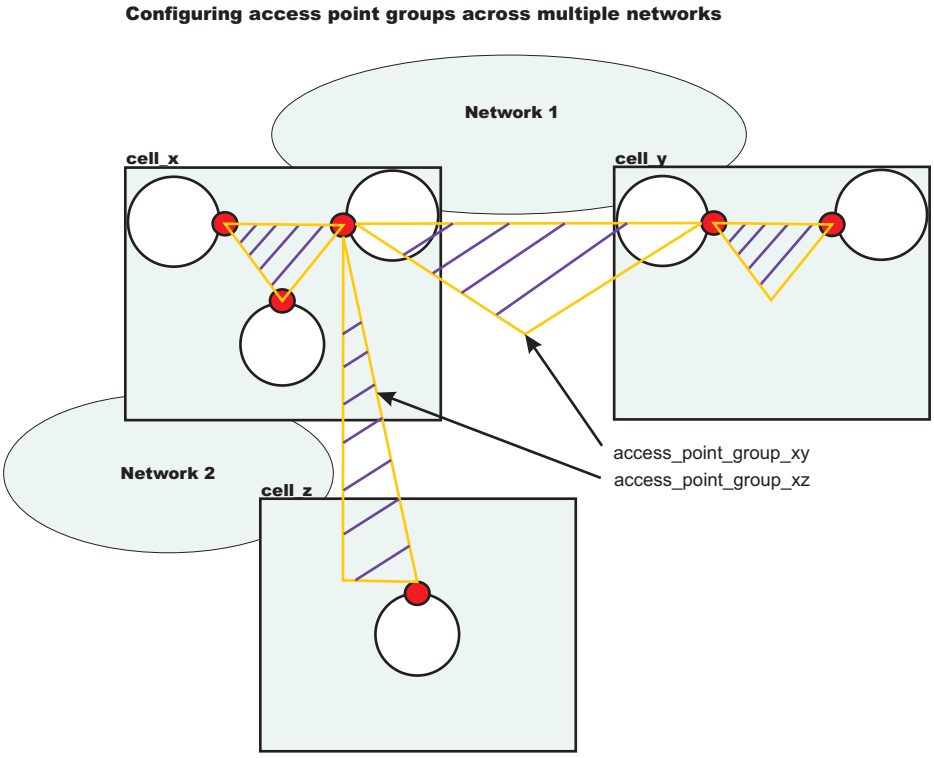


Figure 5. Core group communication across different networks

Communication between core groups using a proxy peer access point

Use a proxy peer when the core groups cannot directly communicate. The two core groups must have access to a single core group that can pass information between the two core groups. To understand what a proxy peer access point does, consider a connecting flight when flying on an airplane. To fly from Pittsburgh to London you first have to fly to New York City, where you change planes and then fly to London. New York City is the *proxy peer access point* for London.

When defining a proxy peer, the *x_core_group_2* core group in the *cell_x* cell cannot communicate directly with the core group in the *cell_z* cell. However, both core groups can communicate with the core group in the *cell_y* cell. To configure communication between the *cell_x* cell and the*cell_z* cell, you must configure two access point groups. The core group access point in the *cell_y* cell is in both the

access_point_group_xy and *access_point_group_yz* access point groups. The following image shows an overview of a proxy peer configuration.

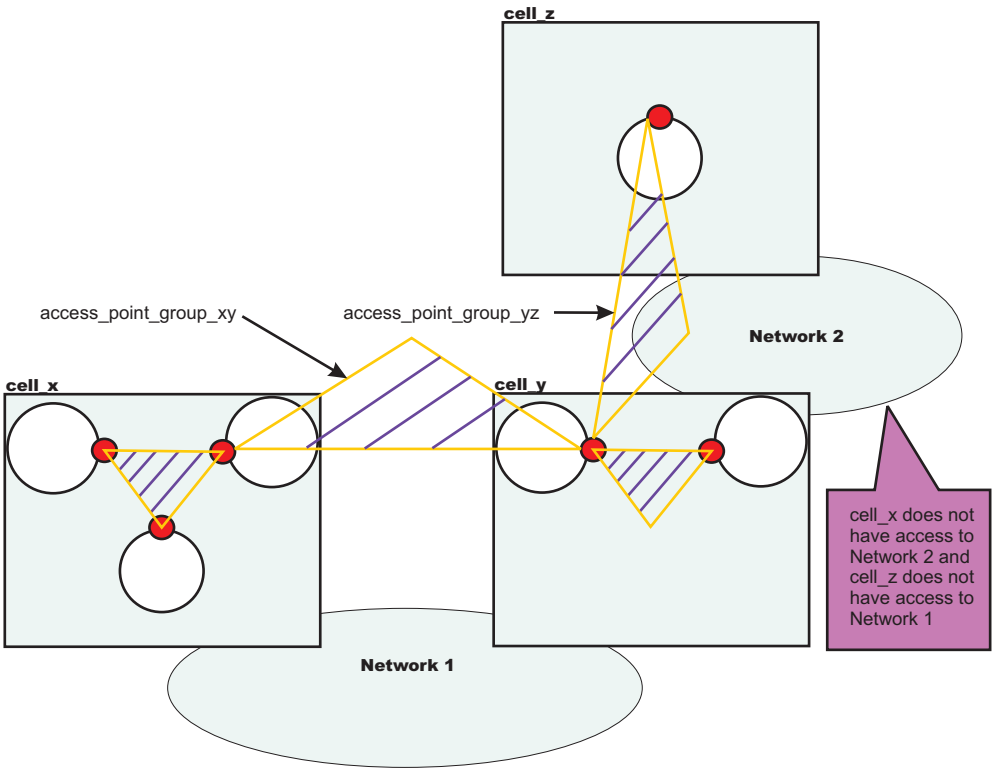


Figure 6. Core group communication using a proxy peer access point

See “Configuring core group communication using a proxy peer access point” on page 423 for more information.

Configuring communication between core groups that are in the same cell

Use this task to configure communication between core groups that are in the same cell.

Configure two core groups with application servers that are in the same cell. For more information about when and how to configure multiple core groups, see “Creating a new core group” on page 387.

You must configure the core group bridge when you have multiple core groups that are in the same cell. Use the core group bridge service to share the availability status of the servers in each core group among all the configured core groups.

To configure communication between core groups, define an access point group. An access point group defines the core groups that can communicate with each other. Each access point group has one core group access point for each core group. Select one or more servers to be core group bridges, and define a bridge interface for each server.

See “Advanced core group bridge configurations” on page 415 for more information about the communication between core groups that are in the same cell.

1. Configure an access point group to define the core groups that need to communicate. An access point group contains the core group access points for the core groups that need to communicate. Core group access points define the set of servers that provide access to the core group. To configure communication between core groups that are in the same cell, you can choose an existing access point group or create a new access point group. To create an access point, complete the following steps:

a. In the administrative console, click **Servers > Core groups > Core group bridge settings > Access point groups > New**.
b. Enter a name for the access point group that is unique within the cell.
c. Add core group access points to your access point group. Choose any available core group access points for the core groups that need to communicate in the cell. A default core group access point is created whenever a core group is created. The access point group that you create must have a core group access point for each core group in the cell.

Restriction: Do not add any peer access points. Add peer access points only if you are configuring communication with a core group in a different cell. If you need to communicate with core groups that are outside of the cell, you must create another access point group that has one core group access point and one or more peer access points. See "Creating advanced core group bridge configurations" on page 414 for more information.

If you use an existing access point group, choose an access point group that does not have peer access points. To configure an existing access point group, perform the following steps:

a. In the administrative console, click **Servers > Core groups > Core group bridge settings**. Your current configuration with any existing access point groups is displayed.
b. Verify that the access point group does not have any peer access points. Peer access point groups are used for communication between core groups in different cells. Click the access point group you want to configure and ensure that no peer access points are listed.
c. Click **Access point groups >** *access_point_group_name* **> Core group access points**.
d. Add core group access points to your access point group. Choose any available core group access points for the core groups that need to communicate. The access point group you create should have a core group access point for each core group in the cell.

2. Create bridge interfaces for each core group access point. The bridge interfaces that you add provide access to the core group. Create at least one bridge interface for each core group access point. To provide high availability for the core group access point, configure two or more bridge interfaces. If a core group has multiple core group access points, each core group access point must contain the same number of bridge interfaces for the same set of servers. To configure bridge interfaces, perform the following steps:
   a. In the administrative console, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Core group access points**.
   b. Click a core group access point in the access point group. Click **Show Detail**.
   c. To create a new bridge interface, click **Bridge interfaces > New**.
   d. Select a node, server, and transport channel chain combination for the bridge interface. Click **OK**. All the bridge interfaces for the core group access points that are in the same access point group must have transport channel chains with the same port name. You can configure the same port name by selecting the same chain name for all of the bridge interfaces. The transport channel chain can be the DCS or DCS-secure channel chains that are created for the DCS_UNICAST_ADDRESS transport chain.
   e. Consider creating at least two bridge interfaces for each access point. If one bridge interface fails, the other can still be active.
   f. Repeat these steps to create bridge interfaces for each core group access point in your access point group.

The core groups that are in the same cell and configured in an access point group can communicate.

In the *cell_x* cell, there are the *x_core_group_1*, *x_core_group_2*, and *x_core_group_3* core groups. Each core group already has a core group access point. The following image illustrates an access point group between the core groups in the *cell_x* cell and an example of the configuration in the administrative console.

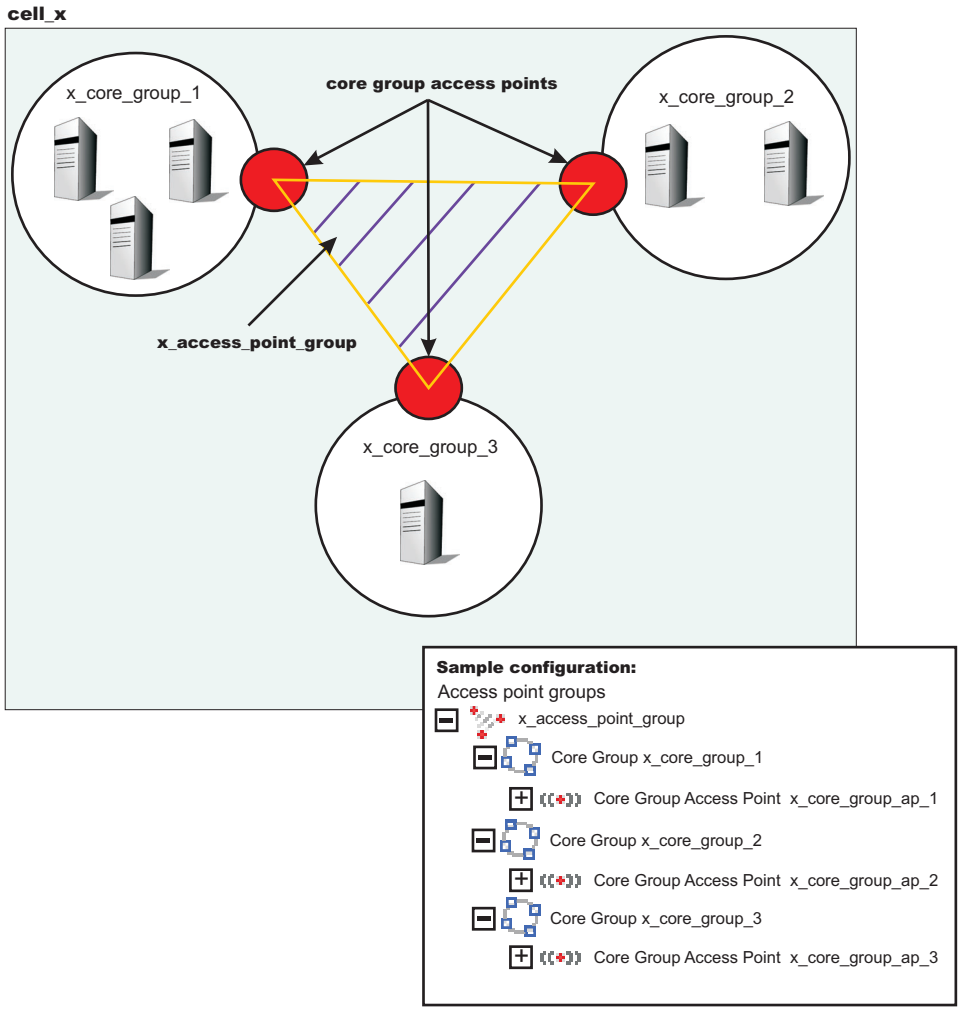


Figure 7. Three core group access points in the same cell belong to the same access point group.

Perform the following steps to configure communication between the three core groups in the *cell_x* cell:

1. Create the x_access_point_group access point group. Add a core group access point to the access point group for each core group that is in the cell. In this example, add the x_core_group_ap_1 , x_core_group_ap_2, and x_core_group_ap_3 access points to the x_access_point_group access point group.
2. Create bridge interfaces for each core group access point. The following diagram illustrates the bridge interfaces for thex_core_group_ap_2 core group access point:

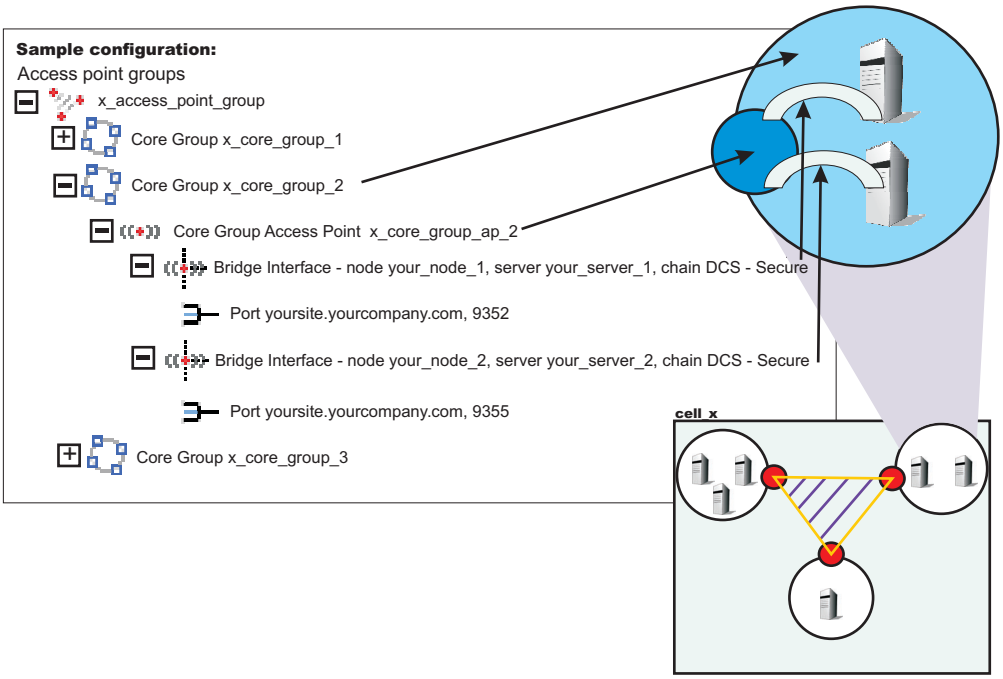


Figure 8. Core group access points contain one or more bridge interfaces.

Create two or more bridge interfaces for each core group access point.

By creating an access point group and adding all core groups in the cell to the access point group, you enabled communication between all the core groups that are in the *cell_x* cell.

You can configure this cell to communicate with core groups in other cells. See “Configuring the core group bridge between core groups that are in different cells” on page 406 and “Configuring core group communication using a proxy peer access point” for more information.

Configuring core group communication using a proxy peer access point

Use this task to configure communication between two core groups when they cannot communicate with each other directly through peer ports.

Configure a proxy peer access point to communicate with a core group if you cannot use peer ports. Use a core group that has configured a peer access point with peer ports to the core group that you want to communicate with. Before completing this task, make sure that you have access to a core group that can communicate with the core group that you cannot communicate with directly. To configure communication between core groups that are in different cells, see “Configuring the core group bridge between core groups that are in different cells” on page 406. If there are multiple core groups in each of the cells, they must be configured to communicate with each other. See “Configuring communication between core groups that are in the same cell” on page 420 for more information.

Use a proxy peer to communicate with a core group that you cannot access directly. This task describes how to configure a proxy peer with three core groups that are each in different cells. The *core_group_x* and *core_group_y* core groups can communicate directly with each other through peer ports. The *core_group_y* and *core_group_z* core groups can also communicate with each other though peer ports. However, the *core_group_x* core group cannot communicate with the *core_group_z* core group. To establish that communication, the *core_group_x* core group has a peer access point that is a proxy peer. The proxy peer is the peer access point to the *core_group_y* core group. For more information, see “Advanced core group bridge configurations” on page 415.

1. Configure the *core_group_x* and *core_group_y* core groups to communicate with each other by creating an access point group. For more information, see “Configuring the core group bridge between core groups that are in different cells” on page 406.
2. Configure the *core_group_y* and *core_group_z* core groups to communicate with each other by creating another access point group. For more information, see “Configuring the core group bridge between core groups that are in different cells” on page 406. When you do this step, create a second access point group in cell 2. The *core_group_2* core group communicates with both the *cell_x* and *cell_z* cells over two different networks.
3. Configure a peer access point that has a proxy peer. Create a new peer access point in the access point group that you created between the *core_group_x* and *core_group_y* core groups.
   a. In the administrative console, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Peer access points**. Create a new peer access point, or select an existing access point.
   b. Enter a unique name for the peer access point. For the other cell, core group, and core group access point values, use the properties of the*core_group_z* core group.
   c. Click **Use a proxy peer access point**. Select the proxy peer access point that is the peer access point that you created in the *cell_x* cell that refers to the core group access point in the *cell_y* cell.

The *core_group_x* core group can communicate with the *core_group_z* core group by using a proxy peer.

The following example shows the configurations in each cell when you configure communication between the *cell_x* and *cell_z* cells using a proxy peer access point:

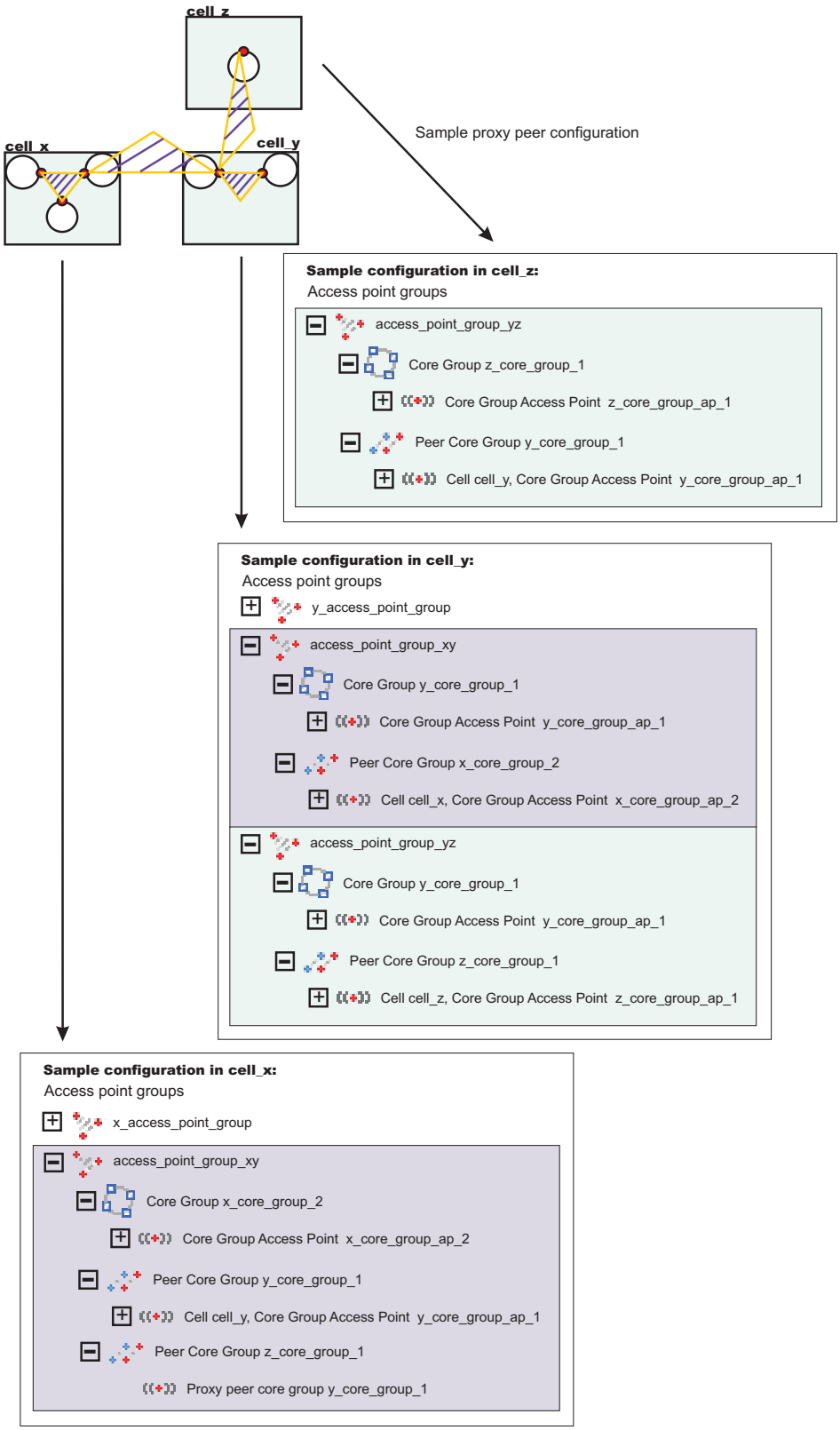


Figure 9. Example proxy peer configuration

By completing this task, you enabled one-way communication between the *core_group_x* and *core_group_z* core groups. If you want to configure communication both ways, you must repeat these steps, configuring a peer access point in the *core_group_z* core group that contains a proxy peer.

Core group bridge custom properties

Use these custom properties for advanced configurations for core groups and core groups that communicate with the core group bridge.

CGB_ENABLE_602_FEATURES

You can define the CGB_ENABLE_602_FEATURES custom property on all of the access point groups in your configuration if you want to be able to add core group bridge servers to the configuration without restarting the other servers in the configuration. After you enable this property, you can add a core group bridge server in one cell, without modifying the configuration in the other cell to include peer ports for the core group bridge server. Instead of manually configuring peer ports in each of the cells, you can configure the peer ports in one cell and let the other cell discover the peer ports for the first cell.

The existence of the CGB_ENABLE_602_FEATURES property enables the property. Therefore, you can set the value to any string value. Setting the value to false does not disable the property. To disable the property, you must remove it from the list of defined custom properties or change its name.

For more information about enabling this property, see “Configuring the core group bridge between core groups that are in different cells” on page 406.

FW_PASSIVE_MEMBER

Use this property in a core group bridge configuration when there is a firewall between the core groups and the secure side of the firewall is configured to listen only.

Set the FW_PASSIVE_MEMBER custom property to make the bridge interfaces that are in the core group access point passive. Set the value on the core group access point that is on the secure side of the firewall so that the core group bridge interfaces listen for connections from the unsecured side of the firewall but do not initiate any connections. The servers on the secure side of the firewall are passive. The custom property should correspond to your defined firewall rules that allow connections from the unsecured region to the secure region only.

To configure this custom property, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Core group access points >** *core_group_access_point_name* **> Show detail > Custom properties > New** in the administrative console.

You also must set this custom property in any peer access points that refer to the core group access points that you configure with this custom property.

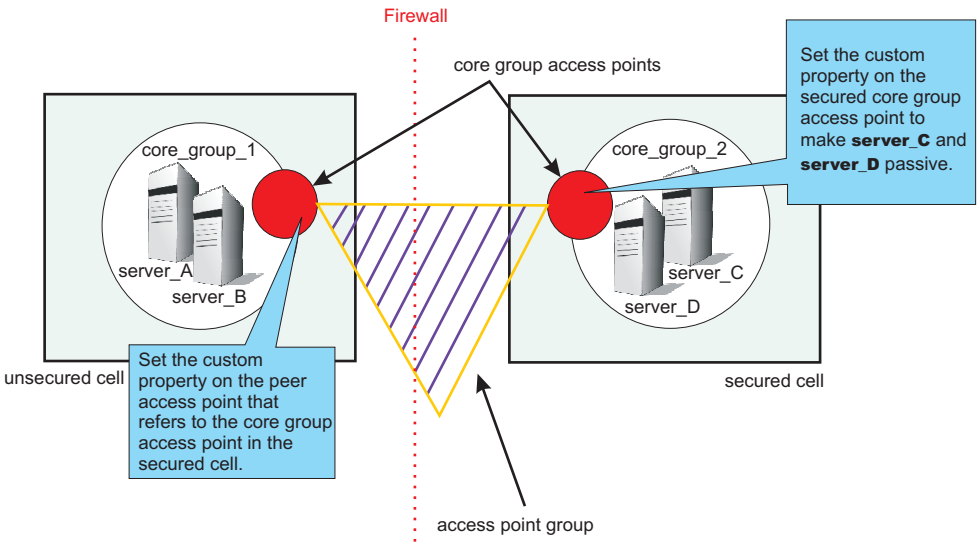


Figure 10. Configuring the FW_PASSIVE_MEMBER custom property

For example, *server_A* and *server_B* are configured in *core_group_1*. *Server_C* and *server_D* are configured in *core_group_2*. *Core_group_2* is behind a firewall that is configured to listen only through the firewall. *Core_group_1* is on the unsecured side of the firewall. *Core_group_1* and *core_group_2* can communicate with each other through an access point group. To configure *server_C* and *server_D* to be passive, perform the following steps:

1. In the administrative console for the cell that contains *core_group_2*, click **Servers** > **Core groups** > **Core group bridge settings** > **Access point groups** > *access_point_group_name* > **Core group access points** > *core_group_access_point_name* > **Show detail** > **Custom properties** >**New** .
2. Add the FW_PASSIVE_MEMBER custom property. Enter any value to enable the property.
3. In the administrative console for the cell that contains *core_group_1*, click **Servers** > **Core groups** > **Core group bridge settings** > **Access point groups** > *access_point_group_name* > **Peer access points** > *peer_access_point_name* > **Show detail** > **Custom properties** > **New**. The peer access point you select should correspond to the core group access point for *core_group_2*.
4. Add the FW_PASSIVE_MEMBER custom property. Enter any value to enable the property.

By configuring the FW_PASSIVE_MEMBER custom property, you configured the servers on the secured side of the firewall, *server_C* and *server_D*, to be passive. These servers listen for connections from the other side of the firewall but do not initiate any connections to the unsecured side of the firewall.

IBM_CS_LS_DATASTACK_MEG

Use this custom property to eliminate a condition that is reported by a message that is displayed repeatedly in your SystemOut.log file.

You might see a message similar to the following message in the SystemOut.log file multiple times:

```
[9/24/04 13:28:19:497 CDT] 00000013 VSync         W
DCSV2005W: DCS Stack DefaultAccessPointGroup.P at Member 172.16.1.86:9353:
The amount of memory available for synchronization is low. The configured memory
size is 418816 bytes. Currently used memory size is 420307 bytes.
```

If the member IP address is in the format of a dotted decimal IP address and port, you can eliminate these messages by increasing the amount of memory that is allocated to the core stack that is used for core group communication. Increase the value of this property until you no longer see the message in your SystemOut.log file. Because the memory is dynamically allocated, setting a larger stack size than you need does not cause memory problems.

Set the custom property on the bridge interface that contains the particular member that is in the messages. You can also set the custom property on the access point group or the core group access point. If you set the value on the access point group or core group access point, all the bridge interfaces that are in the particular group are affected. If you set the value on an individual bridge interface and an access point group or core group access point, the value that is set for the bridge interface is used. If the value is set on both an access point group and a core group access point, the value that is set for the core group access point is used.

To configure this custom property, complete the following steps:

1. Set the custom property in the administrative console.
   - To set the custom property on a bridge interface, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Core group access points >** *core_group_access_point_name* **> Show detail > Bridge interfaces >** *bridge_interface_name* **> Custom properties > New**.
   - To set the custom property on a core group access point, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Core group access points >** *core_group_access_point_name* **> Show detail > Custom properties > New**.
   - To set the custom property on an access point group, click **Servers > Core groups > Core group bridge settings > Access point groups >** *access_point_group_name* **> Custom properties > New**.
2. Add the IBM_CS_LS_DATASTACK_MEG custom property. Enter a value that is greater than the default value of 5 megabytes.

Units	megabytes
Default	5

Troubleshooting high availability environment problems

Review the following topics if you encounter a problem with your high availability environment.

Message HMGR0218I is not displayed after a server starts

In a properly set up high availability environment, a high availability manager can reassess the environment it is managing and accept new components as they are added to the environment. For example, when a Java virtual machine (server) is added to the infrastructure, a discovery process begins. During startup the server tries to contact the other members of the core group. When it finds another running server, it initiates a join process with that server that determines whether or not the server can join the core group. If the new server is accepted as a member of the core group, all of the servers, including the new one, log message HMGR0218I . This message is also displayed on the administrative console.

Message HMGR0218I indicates the number of application servers in the core group that are currently online. If this message is not displayed after a server starts, either a configuration problem or a communication problem has occurred. To fix this situation, verify that the application server is running on a current configuration, by either using the deployment manager to tell the node agent to synchronize, or use the **syncNode** command to manually perform the synchronization. If the server still cannot join the core group, a network configuration problem exists.

CPU starvation messages in SystemOut.log

CPU starvation detected error messages are displayed in the SystemOut.log file whenever there is not enough physical memory available to allow the high availability manager threads to have consistent runtimes. When the CPU is spending the majority of its time trying to load swapped-out processes while processing incoming work, thread starvation might occur. The high availability manager detects this condition, and logs these error messages informing you that threads are not getting the required runtime.

To achieve good performance and avoid receiving these error messages, it is recommended that you allocate at least 512 MB of RAM for each Java process running on a single machine.

High CPU usage in a large cell configuration when security is enabled

With certain configurations and states, the amount of time spent in discovery becomes substantial.

- If a large the number of processes are defined within a core group, a proportionally large number of connections must be established to support these processes.
- If a large number of inactive processes are defined within a core group, a proportionally large number of connections are attempted during each discovery interval.
- If administrative security is enabled, the DCS connections are secured, and the impact of opening a connection greatly increases .

To decrease the CPU time spent in discovery:

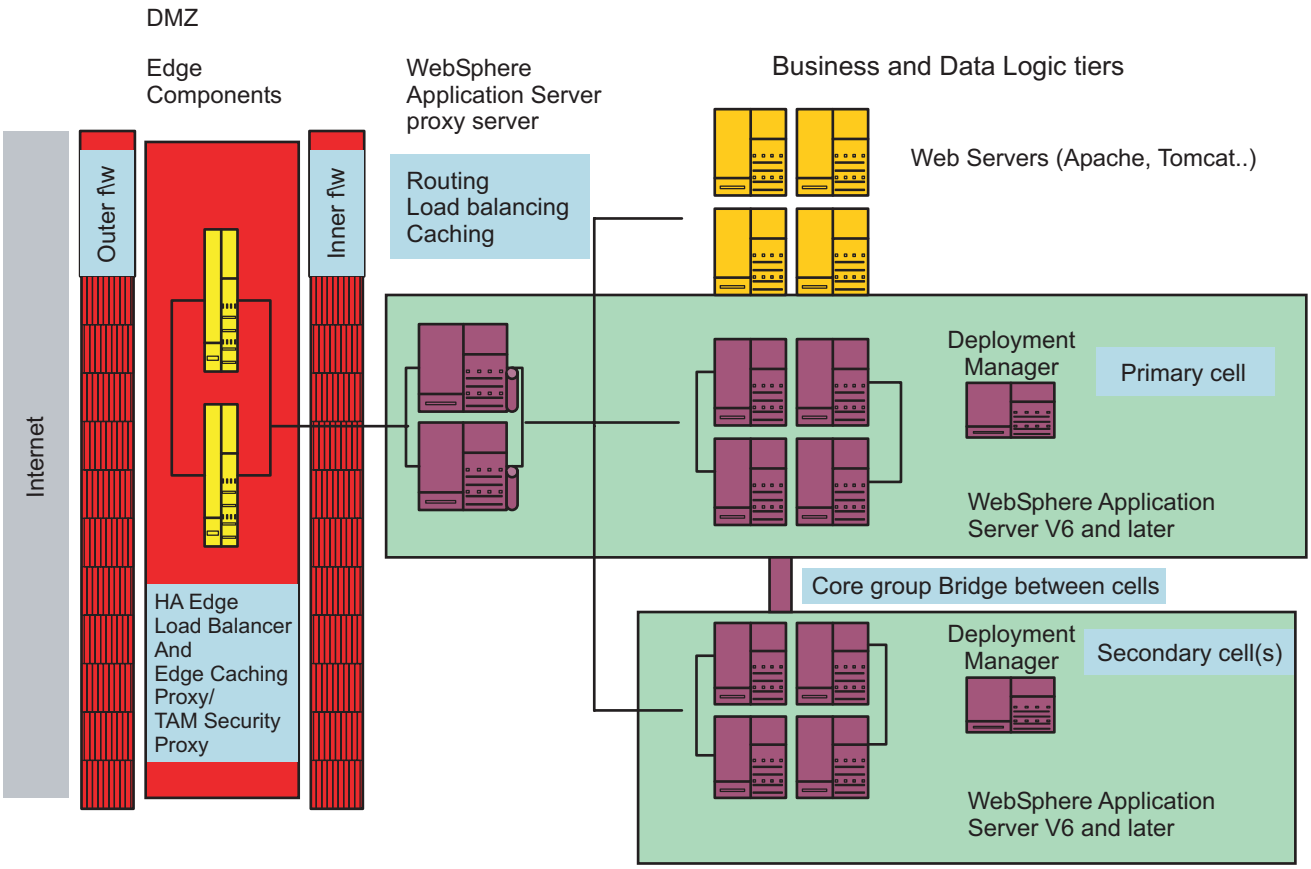
1. In the administrative console, click **Servers > Core groups > Core groups settings** , and then select the -> **DefaultCoreGroup**.
2. Under Additional Properties, click **Custom properties > New**.
3. Enter `IBM_CS_UNICAST_DISCOVERY_INTERVAL_SECS` in the Name field and 120 in the Value field.
4. Click **OK**.
5. Then click **New** again and enter `IBM_CS_SS_SECURE_TOKEN` in the Name field and`false` in the Value field.
6. Click **OK** and then **Save** to apply these changes to the master configuration.
7. Restart the server for these changes to take effect.

Chapter 10. Setting up the proxy server

The *proxy server* is a specific type of application server that routes HTTP requests to content servers that perform the work. The proxy server is the initial point of entry, after the firewall, for requests into the enterprise.

The proxy server acts as a surrogate for content servers within the enterprise. As a surrogate, you can configure the proxy server with rules to route to and load balance the clusters of content servers. The proxy server is also capable of securing the transport, using Secure Sockets Layer (SSL), and the content using various authentication and authorization schemes. Another important feature is its capability to protect the identity of the content servers from the Web clients by using response transformations (URL rewriting). The proxy server can also improve performance by caching content locally and by protecting the content servers from surges in traffic.

A proxy server configuration provides settings that control how a proxy server can provide services for the enterprise applications and their components. This section describes how to create and configure proxy servers in an existing application server environment.



In WebSphere Application Server V6.0.2, you had to augment the deployment manager profile to manage the proxy server. For WebSphere Application Server V6.1 and later versions, the proxy server is managed from the administrative console without initial augmentation.

Creating a proxy server

This topic provides information to create and configure a proxy server.

The proxy server routes requests to application server nodes. The proxy server can dynamically route requests to all on-demand configuration (ODC) enabled application servers without additional configuration.

1. Create a proxy server in the administrative console by clicking **Servers > Proxy Servers**.
2. Click **New**.
3. Select the node on which you want the proxy server to reside. Only Network Deployment nodes display in the selection list. A proxy server can reside only in a Network Deployment node. Enter a name for the new proxy server and click **Next**.
4. Select a proxy server template on which to base your proxy server. Click **Next**. You can select a default template, or you can choose to map to an existing application server.

   Tip:
   Mapping to pre-existing proxy servers is a time-saving technique. You can build one proxy server and apply all of the specific configurations your environment needs, and then use that proxy server as a template.
5. Determine whether or not to generate unique HTTP ports by selecting or clearing Generate unique HTTP ports. Click **Next**. If you create multiple proxy servers on the same node for vertical scaling, then you might select the option to generate unique ports to avoid port conflicts. Certain advanced scenarios pertain to port mapping that might require unique ports. For example, a load balancer can load balance requests to the proxy servers within the same node, assuming that each proxy server is listening on a unique HTTP port. For the proxy server to accept requests for a specific virtual host, it is necessary to add the unique HTTP ports that are generated to the host alias of the virtual host. It might also be necessary to modify the port values that the wizard generates, if these ports conflict with other local servers on the same node.
6. Review the summary panel and click **Finish**.

You now have a functional proxy server that automatically routes HTTP requests to WebSphere Application Server cells. To enable routing to another WebSphere Application Server cell, configure your cell to communicate with other WebSphere Application Server cells.

If the proxy server fails to start when attempting to start it as a non-privileged user on UNIX systems, change the ports of the PROXY_HTTP_ADDRESS and PROXY_HTTPS_ADDRESS transport chains to values greater than `1024`.

Migrating profiles for the WebSphere proxy server

This topic describes how V6.0.2 profiles contain the proxy server feature when migrating to a V6.1 profile.

Migrate from a V6.0.2 profile to a V6.1 profile as follows:

1. Run the **WASPreUpgrade.bat** or the **WASPreUpgrade.sh** command from the *install_root*/bin directory and point it to the V6.0.2 release. This actions makes a copy of all the required V6.0.2 files that are needed for the **WASPostUpgrade.bat** or **WASPostUpgrade.sh** command. You can delete V6.0.2, but this action is not recommended.
2. Create your corresponding profiles in V6.1, if you have not already done so.
3. Run the **WASPostUpgrade.bat** or the **WASPostUpgrade.sh** command from the *install_root*/bin directory and point the commands to the WASPreUpgrade backup directory that you created in step one.

Customizing routing to applications

This topic provides information on disabling routing through the proxy server to applications that are on on-demand configuration (ODC)-compliant application servers.

Follow the steps to disable routing to ODC-compliant application servers. The Web module proxy configuration option will only display if the deployment manager is augmented with the proxy server configuration.

1. From the administrative console, click **Applications > Enterprise Applications**.
2. Select the application you want to customize.
3. Click **Web modules**.
4. Select the Web module from the collection view.
5. Click **Web Module Proxy Configuration**.
6. Deselect **enable proxy** to disable routing using the proxy server.
7. Choose the protocol from the drop-down box for the Web module transport protocol field. Complete this step if you want to use a specific transport protocol, such as HTTP, for the protocol between the proxy server and the application server, rather than the protocol that is used by the client to communicate with the proxy server.

   For example, in order to perform Secure Sockets Layer (SSL) termination, which is also known as SSL offload, for HTTPS traffic for the Web module, select **HTTP** for the transport protocol.

Web module proxy server configuration settings

Use this page to specify proxy server configuration settings for the Web module.

To view this administrative console page, click **Applications > Enterprise Application >** *application_name* **> Manage Modules >** *Web_module_name* **> Web Module Proxy Configuration**.

Name

Specifies the name of the proxy server.

Enable Proxy

Select **Enable Proxy** to enable proxy server to route requests to this Web module. Deselect **Enable Proxy** to disable routing using the proxy server.

Web Module Transport Protocol

Specifies the protocol that the proxy server uses when communicating with this Web module.

Choose the protocol in the Web module transport protocol field if you want to use a specific transport protocol, such as HTTP, for the protocol between the proxy server and the application server. This is an alternative to using the protocol that is used by the client to communicate with the proxy server. For example, in order to perform Secure Sockets Layer (SSL) termination, which is also known as SSL offload, for HTTPS traffic for the Web module, select HTTP for the transport protocol.

Custom Properties

Specifies additional custom properties for this runtime component. Some components use custom configuration properties that are defined by this option.

Routing requests to ODC-compliant application servers in other cells

If you want to isolate proxy servers with firewalls, place them in a cell that is separate from the applications and configure the core group bridge service.

You can configure the proxy server to route requests to applications that are hosted in other cells. The core group bridge service provides communication between a cell with a proxy server and a cell with a target application. Once the cells are linked with core group bridges, the proxy server will be able to route to the application without additional configuration. You can find core group bridge settings by clicking **Servers > Core groups > Core group bridge settings** in the administrative console.

The basic steps to configure cross-cell routing are configuring and enabling core group bridges in the cell that the proxy server belongs to, and in each of the other cells that are being routed to. The core group bridges need to be configured with the same access point group name.

1. Create bridge interfaces, peer access groups and peer ports for cells. The peer access point group name must be the same in the cells. If you want to use CGB_ENABLE_602_FEATURES, enable it on all of the cells involved.
2. Restart all processes that are now bridge interfaces.

If security is enabled, it must be enabled in all cells for the core group bridge service.

Configuring rules to route requests to Web servers

This section provides information to configure rules to route requests to web servers or application servers that are not on-demand configuration (ODC)-compliant.

The proxy server permits explicit configuration of routing rules in order for client requests to route to Web servers or application servers that are not ODC-compliant. This involves the following tasks:

1. Creating a URI group from **Environment > URI groups** in the administrative console to define the URI patterns that are associated with the Web content that is hosted on these servers.
2. Creating a generic server cluster definition from **Servers > Generic server clusters** in the administrative console to define the endpoints that define the cluster membership.
3. Creating routing rules from the detail view of the proxy server panel that associates the inbound virtual host and a defined URI group to a defined generic server cluster.

Modifying the HTTP endpoints that the proxy server listens on

By default, the proxy server listens on the following named endpoints: PROXY_HTTP_ADDRESS and PROXY_HTTPS_ADDRESS. You can modify the ports and hosts that are associated with these endpoints by clicking **Servers > Proxy servers >** *server_name* **> Communications > Ports** in the administrative console.

1. Modify the ports and hosts that are associated with these endpoints by clicking **Servers > Proxy servers >** *server_name* **> Communications > Ports** in the administrative console.
2. Select the port you want to modify.

   The host and port that are associated with the specified endpoints can be modified to suit the requirements of the deployment. The administrator must ensure for correct routing, the virtual hosts that are associated with applications to be routed to are correctly updated with the appropriate host aliases (host aliases with the modified host and port combinations).
3. Click **Apply**.

Restart WebSphere Application Server for this modification to be effective.

Adding a new HTTP endpoint for the proxy server

An administrator can use the proxy server to create additional endpoints to listen for HTTP(S) requests.

The following steps will create a new transport chain:

1. Click **Servers > Proxy servers >** *<server_name>* **HTTP proxy server settings > Proxy server transports** in the administrative console.
2. Click **New** to launch the Create new transport chain wizard.
3. Determine a name for the transport chain.

4. Choose a template based on whether the endpoint should accept HTTP (proxy template) or HTTPS (proxy-secure template) requests. The wizard prompts for the host and port for the endpoint. Creating a new endpoint for the proxy server requires a restart of the proxy server to be effective.

Setting up a custom SSL repertoire

This topic provides information to set up a custom Secure Sockets Layer (SSL) repertoire for inbound (client to proxy server) HTTPS traffic.

Follow the steps to create a new SSL repertoire that contains a valid SSL certificate:

1. In the administrative console, click **Security > SSL** to create a new SSL repertoire. See for more information on SSL repertoire settings.
2. Click **Servers > Proxy servers** *<server_name>* **> HTTP proxy server settings > Proxy server transports**.
3. Select the HTTPS transport chain to customize.
4. Click **SSL inbound channel**.
5. From the SSL repertoire drop-down menu, select the SSL repertoire that you created in step one.
6. Click **Apply**.

Caching content in the proxy server

This topic provides information on caching static and dynamic content in the proxy server.

Setting up caching in the proxy server

The proxy server provides the capability to cache and retrieve responses locally. It uses the WebSphere object cache instance as the store for the responses. Each proxy server has a default object cache instance that it uses for storage and retrieval of cached responses.

To configure the object cache instance for size, disk offload location, and other such capabilities, in the administrative console, click **Servers > Proxy servers >** *server_name* **> HTTP proxy server settings > Proxy cache instance**. Then select the proxy cache store instance and enable configuration attributes such as cache size, disk offload, and cache replication.

For disk offload, it is recommended that the location be set to a dedicated disk partition. To enable caching at the proxy server, in the administrative console, click **Servers > Proxy servers >** *server_name* **> HTTP proxy server settings > Proxy settings** page in the administrative console. Then select **Enable caching** and choose a cache instance from the drop-down box.

Caching static content

Static content is Web content that is public and accompanied by HTTP response headers, such as EXPIRES and LAST_MODIFIED_TIME, that describe how long the response can be cached. The proxy server uses the HTTP 1.1 RFC (2616), which specifies how content should be treated and includes capabilities such as VARY header support for caching variants of the same resource Uniform Resource Identifier (URI).

Static caching is enabled by default when caching is enabled for the proxy server.

Caching dynamic content

Dynamic content is content that an application, that is hosted on an application server, generates. A proxy server caches dynamic content only if the content is identified as edge cacheable in the cachespec.xml file for the application. All of the information that describes the cache, such as the ID to use for the cache,

dependency identifiers for invalidation, and expiration times, is also defined in the cachespec.xml file. Proxy Server uses the ESI protocol to obtain this information from the file.

See the *Administering applications and their environment* PDF for more information on how to set up a cachespec.xml file for an application.

Cached dynamic content can be invalidated by events in the application server. The ESI Invalidation Servlet, that is contained in the DynacacheEsi.ear application, propagates these invalidation events from the application server to the proxy server. The DynacacheEsi.ear is shipped with WebSphere Application Server and must be deployed in the cluster with the application that is generating the dynamic content for dynamic caching at the proxy server to function properly.

You enable cacheablity and invalidation of dynamic content when you enable servlet caching on the application server, and specifying the cache criteria in a cachespec.xml file that is associated with that application. To enable dynamic content to be cacheable with the proxy server, in the administrative console, click **Servers > Proxy servers >** *server_name* **> HTTP proxy server settings > Proxy settings**, and then select **Cache dynamic content**. Invalidations are received by connecting to the cache update URI that is associated with the invalidation servlet hosted on the application server cluster.

Routing requests from a plug-in to a proxy server

This topic provides information on setting up a WebSphere Application Server plug-in to route requests to a proxy server.

An administrator may choose to set up a Web server, such as IBM HTTP Server, with the WebSphere Application Server plug-in as a front-end to the proxy server. The plug-in configuration file for such a topology cannot use the traditional plug-in configuration generation mechanism if the requests are routed through the proxy server.

To generate the `plugin-cfg.xml` file to use with the Web server plug-in to route through the proxy server, complete the following steps:

1. From the administrative console, click **Servers > Proxy servers >** *server_name* **> HTTP proxy server settings > Proxy settings**.
2. In the **Generate plug-in configuration** drop-down menu, select the cell scope. This generates the `plugin-cfg.xml` file for all proxy servers in the cell. You will find the `plugin-cfg.xml` in the <*install_root*>/<*profile_dir*>/`etc` directory. Select node or server for a smaller scope.
3. **Optional:** If you have a script that manually copies the `plugin-cfg.xml` file from the node to the plug-in installation location, enter the path to the script in the **Plugin config change script** field.
4. In the Trusted Security Proxy field, add the hostname or IP address of the node for the plug-in that serves as the trusted intermediary for the proxy server.
5. Click **OK**.
6. Disable the automatic propagation of the plug-in if you are using IBM HTTP Server with remote administration. From the administrative console, click **Servers > Web servers >** *server_name* **> Plug-in properties**. Deselect **Automatically propagate plugin configuration file**. This will prevent WebSphere Application Server from copying the traditional `plugin-cfg.xml` file over the proxy server `plugin-cfg.xml` file.
7. Save your changes.
8. Stop and restart the proxy server. The `plugin-cfg.xml` file will be in the <*install_root*>/<*profile_dir*>/`etc/plugin-cfg.xml` directory for the node. If you do not have a script in the **Plugin config change script** field, manually copy the `plugin-cfg.xml` file to the plug-in.

To ensure that the proxy server trusts the Web server, add the host name or address of the Web server to the Trusted security proxies section on the Proxy Settings panel of the WebSphere Application Server

administrative console (**Servers > Proxy servers >** *server_name* **> HTTP proxy server settings > Proxy settings**). This enables the proxy server to honor the WebSphere Application Server private headers that are set by the fronting intermediary server.

Creating a proxy server cluster

This topic describes how to create a cluster of proxy servers, using the **wsadmin** command, that can route requests to applications in a cell.

The cluster includes the machines and nodes that are going to belong to the proxy server cluster. You need to have WebSphere Application Server V6.1 installed. Consider how requests are routed to the proxy server cluster (for example, using domain name server (DNS), Load balancer, or the proxy server). Before you create a cluster, start the deployment manager.

Create a proxy server cluster, as follows:

1. Start the wsadmin utility.
2. Create an empty cluster with no members, by typing:

   ```
   $AdminTask createCluster { -clusterConfig {{<name_of_cluster> true PROXY_SERVER}}}
   ```

   Or, to create a cluster and add a specified proxy server to that cluster, type:

   ```
   $AdminTask createCluster { -clusterConfig {{<name_of_cluster> true PROXY_SERVER}} -convertServer
        {{<node_name> <name_of_proxy_server> "" "" ""}}}
   ```

   This proxy server serves as the template for all subsequent members that are added to the cluster.
3. Add one member at a time to the cluster, as follows:

   ```
   $AdminTask createClusterMember {-clusterName <name_of_cluster -memberConfig
         {-memberNode > <node_name> -memberName <name_of_proxy_server>}}
   ```

   If no members exist in the cluster, the first member that is added serves as the template for subsequent members that are added to the cluster.

   When a proxy server is added to a cluster, proxy-specific configuration settings for it can only be configured using the wsadmin scripting client.
4. Save the configuration changes, as follows:

   ```
   $AdminConfig save
   ```
5. Start the proxy server cluster so that request routing is enabled, as follows:

   ```
   $AdminTask startCluster <name_of_cluster>
   ```
6. Configure requests to route to the proxy server. For DNS-based routing, associate the logical name of the site with the IP addresses of the proxy server cluster members in DNS.

   For Load balancer routing, configure the IP addresses of the cluster members as the target of the virtual cluster.

   For Edge proxy or IBM HTTP Server with WebSphere Application Server plug-in-based routing, generate the plug-in configuration file for the proxy server cluster, and configure the Edge proxy or the WebSphere Application Server plug-in with this information.

The proxy server cluster is created with the members and is enabled for routing traffic.

Monitor the traffic. See “Monitoring the proxy server with PMI” on page 438 for more information.

Monitoring the proxy server with PMI

You can monitor the traffic of a proxy server using the Performance Monitoring Infrastructure (PMI) function.

The proxy server must be running before performing these steps.

1. In the administrative console, click **Monitoring and Tuning > Performance Monitoring Infrastructure (PMI)** in the console navigation tree. The proxy server collection page displays.
2. Select the proxy server from the collection list.
3. Click **Custom**. The Custom monitoring level panel displays.
4. In the navigation tree, expand **Proxy Module**. Select the statistics you want to view, then click **Enable**.

Monitoring traffic through the proxy server

You can monitor traffic, such as requests and connection statistics, through the proxy server.

You should know the machines and nodes that will belong to the proxy server cluster before completing these steps. Also, WebSphere Application Server V6.1 needs to be installed on those machines.

1. Ensure that the proxy server is running and there is some traffic flowing through the proxy server.
2. Obtain the proxy server MBean and invoke the operation to get the route statistics as follows:
   a. Start **wsadmin**.
   b. Get all of the traffic statistics through the proxy server using

```
$AdminControl queryName type=ProxyServer,*
set proxymbean <cut and paste the MBean identifier from the previous command output>
$AdminControl getAttribute $proxymbean stats
```

   The **$AdminControl queryName** command lists all of the proxy server MBeans. There will be one per active proxy server in the cell. Set the *proxymbean* variable to the appropriate proxy server MBean from the output of the previous command.

The traffic that flows through the proxy server can now be monitored.

Static content caching in the proxy server

The security proxy, or the intermediary that is the entry point into the enterprise, is responsible for terminating SSL connections with the client, authenticating the request, propagating the connection characteristics for the client, and any other credentials to the application server in the enterprise.

The proxy server enables security proxies to be identified so that private headers that are set in the request are propagated as they are to the application servers. Identify the list of security proxies that are trusted by the proxy server using the trusted security proxies field. To access this administrative console field, click **Servers > Proxy servers >** *<server_name>* **> HTTP proxy server settings > Proxy settings**. You can find trusted security proxies in the **Security** section of this administrative console page.

Overview of the custom error page policy

The custom error page policy is a feature that enables the proxy server to use an application to generate an HTTP error response. With this capability, the administrator can return a polished error page when the proxy server generates an error, or when a content server returns an unsuccessful response.

The following action describes scenarios for how the error page policy is used when it is configured:

- **Internal error**
  1. The client sends the following request to the proxy server: `GET /house/rooms/kitchen.jpg HTTP/1.1`.

2. The proxy server generates an internal error because no servers map to the request (HTTP 404 – File not found).
3. The error policy is configured to handle HTTP 404 responses, so it sends a request to the error page application to retrieve error content to send to the client. The request URI and HTTP response code are included as query parameters in the request to the error page application. If the configured error page application URI is /ErrorPageApp/ErrorPage, then the request URI to the error page application is: `/ErrorPageApp/ErrorPage?responseCode=404&uri=/house/rooms/kitchen.jpg`. The query parameters “responseCode” and “uri” are sent to the error page application by default.
4. The proxy server returns an HTTP 404 response with content from the error page application.

- **Remote error**
  1. The client sends the following request to the proxy server: `GET /house/rooms/kitchen.jpg HTTP/1.1`
  2. The proxy server forwards the request to the `homeserver.companyx.com` content server.
  3. The `homeserver.companyx.com` content server is unable to locate the `/house/rooms/kitchen.jpg` file and sends an HTTP 404 response (File not found) to the proxy server.
  4. The error policy is configured to handle HTTP 404 responses, so it sends a request to the error page application to retrieve error content to send to the client. The request URI and HTTP response code are included as query parameters in the request to the error page application. If the configured error page application URI is `/ErrorPageApp/ErrorPage`, then the request URI to the error page application is: `/ErrorPageApp/ErrorPage?responseCode=404&uri=/house/rooms/kitchen.jpg`. The query parameters “responseCode” and “uri” are sent to the error page application by default.
  5. The proxy server returns an HTTP 404 response with content from the error page application.

A sample error application is available in the *<WAS_INSTALL_ROOT>*`/installableApps/HttpErrorHandler.ear` file.

On Demand Configuration

On Demand Configuration (ODC) enables WebSphere Application Server components, such as the proxy server, to build WebSphere Application Server application deployment, availability, and accessibility information in order to route requests for these applications without any additional configuration at the proxy server.

ODC uses WebSphere Application Server high availability capabilities to publish and subscribe updates. The deployment manager within a cell and the application servers that have the ODC capabilities typically publish the updates that are subscribed to by intermediaries such as the proxy server.

ODC is supported on WebSphere Extended Deployment V5.1 and on WebSphere Application Server V6.0 and higher versions of WebSphere Application Server Network Deployment.

Request mapping

This topic provides an overview of how the proxy server matches an HTTP request that is received to an application that is deployed in the cell or routing rule.

Unlike the Apache Web server or Caching Proxy, which have flat configuration files with routing precedence that is inherent to the ordering of directives, the proxy server uses a best match mechanism to determine the installed application or routing rule that corresponds to a request. The virtual host or URI patterns determine the best match for a Web module or routing rule. For applications that are deployed in clusters, the proxy server maintains affinity (Secure Sockets Layer ID, cookie, and URL rewriting), otherwise, a weighted round-robin approach is used to select the target server. The following examples address various routing scenarios for when routing rules and applications are deployed within the same cell.

Proxy environment. A WebSphere Application Server proxy called `proxy1` is active in the same cell as the applications and routing rules. All of the applications and routing rules are enabled in the cell for `proxy1`, and PROXY_HTTP_ADDRESS for proxy1 is set to `80`.

Virtual host	Host name	Port
default_host	host1.company.com	80
	host1.company.com	9080
	*	80
proxy_host	host2.company.com	80
	*	443
	*	80
server_host	host3.company.com	80

URI group name	URI patterns
ALL	/*
ROOMS	/kitchen/*, /bathroom/*, /bedroom/*
CONFLICT	/WM2C/*

Generic server cluster name	Protocol	Host	Port
CLUSTER1	HTTP	webserver1.company.com	9081
		webserver2.company.com	9083
CLUSTER2	HTTP	host47.company.com	8088
		host48.company.com	8088
CLUSTER2-SSL	HTTPS	host47.company.com	8443
		host48.company.com	8443

Routing rule name	Virtual host	URI group	Action
ALLTOCLUSTER1	proxy_host	ALL	Generic server cluster - CLUSTER1
ROOMTOCLUSTER2	proxy_host	ROOMS	Generic server cluster - CLUSTER2
ALLTOCLUSTER2	server_host	ALL	Generic server cluster - CLUSTER2
REDIRECTTOCONFLICT	default_host	CONFLICT	Redirect - http://www.conflict.com

Application name	Context root	Web module name	Virtual host	Web module URI patterns
App1	/WM1A/	Web Mod A	default_host	wm1a.jsp
	/WM1B/	Web Mod B	default_host	wm1b.jsp
App2	/WM2C/	Web Mod C	default_host	/*, wm2c.jsp
	/WM2D/	Web Mod D	default_host	/*, wm2d.jsp

Example 1: Basic request. The `proxy1` proxy receives the following request:

```
GET /WM1A/wm1a.jsp HTTP/1.1
Host: host1.company.com
```

Result. The wm1a.jsp file is sent as the response. The ALLTOCLUSTER1 routing rule is a possible match, but Web Mod A is chosen as the best match by proxy1 because the combination of its context root and URI pattern /WM1A/wm1a.jsp is a better match than /*. Web Mod A is also chosen as the best match because its virtual host contains the host1.company.com:80 alias, which is a more specific match than the *:80 wild card alias.

Example 2: Routing rules that use the same URI group and different virtual hosts . The proxy1 proxy receives the following request:

```
GET /index.html HTTP/1.1
Host: host3.company.com
```

Result. The proxy1 proxy maps the request to the ALLTOCLUSTER2 routing rule, and a response is received from a server in CLUSTER2. The ALLTOCLUSTER1 routing rule is a possible match and can handle the request if the ALLTOCLUSTER2 routing rule did not exist. However, the ALLTOCLUSTER2 rule is the best match because its virtual host (server_host) explicitly lists host3.company.com.

Example 3: Routing rules that use same virtual host and different URI groups. The proxy1 proxy receives the following request:

```
GET /kitchen/sink.gif HTTP/1.1
Host: host2.company.com
```

Result. The proxy1 proxy maps the request to the ROOMSTOCLUSTER2 routing rule and a server from the CLUSTER2 cluster sends a response. The ALLTOCLUSTER1 routing rule is a possible match, but the ROOMSTOCLUSTER2 rule is the best match because its URI group contains a pattern /kitchen/* that is a better match for the request URI /kitchen/sink.gif.

Example 4: Routing rule URI group conflicts with URI pattern of a Web module that uses the same virtual host. The proxy1 proxy receives the following request:

```
GET /WM2C/index.html HTTP/1.1
Host: host1.company.com
```

Result. Indeterminate. It is unknown whether Web Mod C or the REDIRECTTOCONFLICT routing rule handles the request because they use the same virtual host and have the same URI pattern. In such cases, the ID DWCT0007E message is displayed in SystemOut.log file for the proxy1 proxy. In this example, changing the REDIRECTTOCONFLICT routing rule to use a different virtual host resolves the problem.

Example 5: The PROXY_HTTP_ADDRESS address is not in the virtual host. Assume that the proxy1 proxy address, PROXY_HTTP_ADDRESS, is changed to 81, while all of the other configuration information remains the same. The proxy1 proxy receives the following request:

```
GET /index.html HTTP/1.1
Host: host1.company.com:81
```

Result. The proxy1 proxy is unable to handle the request because the PROXY_HTTP_ADDRESS address is not available in a virtual host and will send an HTTP 404 response back to the client.

Session failover in the proxy server

This article describes how the proxy server handles session failover.

You can enable memory-to-memory replication on an application server to maintain session state in multiple servers. In this case, a private header is added to the response which identifies the backup servers for the session. The proxy server reads this header and maintains a list of backup servers for a

session. If the proxy server fails to route to the primary server, it tries to route to the backup servers. If none of the backup servers are available, a deterministic algorithm selects one of the available servers; consequently, multiple proxy servers will route to the same server.

If the set of servers that are hosting a session changes, the private response header causes the proxy server to update its list of servers for the session. It is possible that the set of servers is updated, but the proxy server has not yet received an updated response header. In this case, the proxy server routes to a server that does not contain the session data. If this occurs, the backend server obtains the session data from a server that contains the session data. There is no functional difference in this case; however, there is a performance difference due to the cost of obtaining the session data from another server.

Session Initiation Protocol proxy server

The Session Initiation Protocol (SIP) proxy server for WebSphere Application Server is used to initiate communication and data sessions between users. It delivers a high performance SIP proxy capability that you can use at the edge of the network to route, load balance, and improve response times for SIP dialogs to back-end SIP resources. The SIP proxy provides a mechanism for other components to extend the base function and support additional deployment scenarios.

The SIP proxy design is based on the WebSphere Application Server HTTP proxy architecture and can be considered a peer to the HTTP proxy. Both the SIP and the HTTP proxy are designed to run within the same WebSphere Application Server proxy server and both rely on a similar filter-based architecture for message processing and routing.

A SIP proxy serves as the initial point of entry, after the firewall, for SIP messages that flow into and out of the enterprise. The SIP proxy acts as a surrogate for SIP application servers within the enterprise. In fact, the nodes that host the SIP proxy servers host the public SIP domain of the enterprise. As a surrogate, you can configure the SIP proxy with rules to route to and load balance the clusters of SIP containers. The SIP proxy is capable of securing the transport, using secure sockets layer (SSL), and the content, using various authentication and authorization schemes.

The SIP proxy is also responsible for establishing outbound connections to remote domains on behalf of the back-end SIP containers and clients that reside within the domain that is hosted by the proxy. Another important feature of the SIP proxy is its capability to protect the identity of the back-end SIP containers from the SIP clients.

Installing a Session Initiation Protocol proxy server

This topic provides information to install a Session Initiation Protocol (SIP) proxy server.

Ensure that you have a SIP application installed. The SIP container will not listen on a port without a SIP application installed.

Install a SIP proxy server. If you created a cell, deployment manager, and node when you installed WebSphere Application Server, the first five steps do not apply and you can skip to step six.

1. Run the profile creation wizard and create a profile.
2. Run the profile creation wizard and create a deployment manager profile.
3. Start the deployment manager.
4. Federate the node that contains your newly-created profile into the deployment manager cell. You can federate the node in one of the following ways:
   - Type *<WAS_home>*profiles/*<name_of_profile>*/bin/addNode *<deployment_manager_host_name>* 8879

     8879 is the default bootstrap port. The server that is being federated should not be running when completing this task from a command line.
   - From the deployment manager administrative console, click **Application Servers > Nodes** and adding the node.

The server in the node that is being federated needs to be running when federating through the deployment manager administrative console.

5. Create a cluster in the administrative console by clicking **Servers > Clusters > New**. Add the federated server to the cluster.

   Important: Be sure that the default cluster that is defined in the SIP proxy settings points to a valid cluster. Do not use **none**, which is the default.

6. Create a proxy server on the node that you just federated by clicking **Servers > Proxy Server > New**.

   Important: Be sure to select SIP protocol.

You now have a functional SIP proxy server.

Tip: The virtual host for your SIP container ports must be defined, and the SIP container(s) must be restarted after adding the new virtual host.

Communicating with external domains

The general approach for providing secure communications between two independent domains or communities (each maintaining distinct directories) relies on *identity assertion*, where a trust relationship is established between two distinct domains using a certificate exchange during the setup of the physical Secure Sockets Layer (SSL) connection between the two domains.

Authentication of Session Initiation Protocol (SIP) messages that are sent by end users needs to occur only in the local domain for the user. All user messages traverse through the SIP container local domain before being sent on to the remote domain. If a message is received from a remote domain over a secured connection that is mutually authenticated in the manner described as follows, it is assumed that the message is authenticated by the remote domain because of the trust relationship. An administrator can enable support for external domains in the SIP proxy as follows:

1. Enable client authentication within the SSL repertoire that is assigned to all the inbound channel chains (or endpoints) that are to receive inbound connections from remote domains.
2. Ensure that all trusted certificate authorities are set up in the trust store that are assigned to the SSL repertoires mentioned in the previous step. Set up the asymmetric key pair (public and private keys) for the local domain, with the proper chain of certificates that are associated with the local domain.
3. Configure all the distinguished names (DNs) that are associated with the remote domains to support. The DN is part of the X.509 certificate that is sent by the remote domain server when the SSL connection is set up. Within the configuration model, each SIP external domain entry includes a field for the remote DN.
4. Assuming that the SIP infrastructure is deployed within each independent domain, provide the DN to the remote domain administrator that is included in local domains public certificate. With this action, the remote domain administrator can configure the proper external domain DN.

   With this approach, the Java Secure Socket Extension (JSSE) is responsible for authorizing the certificate that is received over a new inbound connection from a remote domain. This authorization is based on the agreed upon certificate authorities whose certificates are set up in the local trust store. If the remote domain certificate is authorized, it is then the responsibility of the SIP proxy to filter the connections, based on the DN that is associated with the remote domain certificate. The proxy also validates outbound connections by ensuring that the DN that is received in the remote server certificate matches the DN configured for the remote external domain.

   The SIP proxy must recognize when identity assertion is in use so that it can inform the SIP container that no message authentication is required over this mutually authenticated connection. This communication is done by adding the P-Preferred-Identity SIP header, which is described in RFC 3325, in all SIP messages that are sent from the proxy to the SIP container that arrive over the authenticated connection. The SIP container only recognizes this header when it is received from a device that resides in the trusted domain, specifically the SIP proxy. It is up to the SIP proxy to remove

this header from any inbound messages that are received over any connections to remote devices that are not considered part of the trusted domain. You can also use this header to support the addition of proxy authentication.

Tracing a Session Initiation Protocol proxy server

You can trace a Session Initiation Protocol (SIP) proxy server, starting either immediately or after the next server startup.

To trace a SIP proxy server, complete the following steps:

1. Start WebSphere Application Server Network Deployment, and open the administrative console.
2. In the administrative console, click **Troubleshooting** → **Logs and trace**.
3. Select the name of the server for the SIP proxy server.
4. From the General Properties section, click **Diagnostic Trace Service**.
5. Select one of the following options:

Option	Description
Configuration	To start tracing after the next server startup
Runtime	To start tracing immediately

6. Replace the content of the trace specification with the following code: `com.ibm.ws.proxy and com.ibm.ws.sip`.
7. Make sure that the **Enable trace with following specification** check box is checked.
8. Click **Apply** → **Save**.

When the changes take effect, SIP proxy server tracing messages display in *WASProductDir*/logs/*serverName*/trace.log on the SIP proxy server node, where *WASProductDir* is the fully-qualified path name of the directory in which WebSphere Application Server is installed and *serverName* is the name of the specific instance of the application server that is running the SIP proxy server to be traced.

SIP proxy settings

The SIP proxy settings page contains general configuration items that affect outbound transport configuration, toleration of IP Sprayer devices, and access logging configuration.

To view this administrative console page, click **Servers > Proxy Servers >** *server_name* **> SIP proxy settings**

On the **Configuration** tab, you can edit SIP proxy settings fields.

Default cluster

This settings typically defaults to null and it must be set to a valid cluster before any SIP messages are routed thought the proxy server. The default cluster indicates which cluster of application servers should receive SIP traffic when there are no cluster selection rules defined, or when none of the existing cluster selection rules match.

Enable TCP sprayer

Enables and disables SIP outbound request rewriting that enables the SIP proxy to operate behind an IP sprayer.

When this control is checked, TCP host and TCP port are enabled. When this control is unchecked, TCP host and TCP port are disabled.

Data type Boolean
Default false

TCP host

This field contains the host name that the SIP proxy writes outbound requests to so that the receiving agent connects back to the IP sprayer.

Enabled only when Enable TCP sprayer is checked.

Data type	String
Default	Blank

TCP port

This field contains the port that the SIP proxy writes outbound requests to so that the receiving agent connects back to the IP sprayer.

Enabled only when enable TCP sprayer is checked.

Range	1 to 65535
Data type	Integer
Default	Blank

Enable SSL sprayer

Enables and disables SIP outbound request rewriting that enables the SIP proxy to operate behind an SSL sprayer.

When this control is checked, SSL host and SSL port are enabled. When this control is unchecked, SSL host and SSL port are disabled.

Data type	Boolean
Default	false

SSL host

This field contains the host name that the SIP proxy writes outbound requests to so that the receiving agent connects back to the SSL sprayer.

Enabled only when Enable SSL sprayer is checked.

Data type	String
Default	Blank

SSL port

This field contains the port that the SIP proxy writes outbound requests to so that the receiving agent connects back to the SSL sprayer.

Enabled only when enable SSL sprayer is checked.

Range	1 to 65535
Data type	Integer
Default	Blank

Enable UDP sprayer

Enables and disables SIP outbound request rewriting that enables the SIP proxy to operate behind a UDP Sprayer.

When this control is checked, UDP host and UDP port are enabled. When this control is unchecked, UDP host and UDP port are disabled.

Data type	Boolean
Default	false

UDP host

This field contains the host name that the SIP proxy writes outbound requests to so that the receiving agent connects back to the UDP sprayer.

Enabled only when Enable UDP sprayer is checked.

Data type	String
Default	Blank

UDP port

This field contains the port that the SIP proxy writes outbound requests to so that the receiving agent connects back to the UDP sprayer.

Enabled only when enable UDP sprayer is checked.

Range	1 to 65535
Data type	Integer
Default	Blank

Enable access logging

This field enables and disables access logging.

When this control is checked, access log maximum size and proxy access log is enabled. When this control is unchecked, access log maximum size and proxy access log is disabled.

Data type	Boolean
Default	Unchecked (false)

Access log maximum size

This field indicates the maximum size, in megabytes, of the access log before it rolls over.

Enabled only when enable access logging is checked.

Range	1 to 65535
Data type	Integer
Default	20

Proxy access log

This field indicates the location of the SIP proxy access log.

Enabled only when enable access logging is checked.

Range	Must be a valid path name
Data type	String
Default	`$(SERVER_LOG_ROOT)/sipproxy.log`

SIP external domains collection

The external domain collection panel provides create, remove and update capabilities for the external domain routing configuration.

To view this administrative console page, click **Servers > Proxy Servers >** *server_name* **> SIP proxy settings > External Domains**

New

Create a new external domain entry. Clicking **New** launches the External Domain Detail panel.

Delete

Used to remove an external domain entry.

Domain

The domain string that is mapped to the associated protocol, host, and port configuration.

Distinguished name

The name that is associated with the external domain. It is used when SSL client authentication is enabled to limit connections from an external domain.

Protocol

This field contains the host name that the SIP Proxy will write to outbound requests so that the receiving agent will connect back to the IP Sprayer.

Host

The host that will be used to make the SIP connection associated with the domain.

Port

The port that is used make the SIP connection associated with the domain.

SIP external domains

The external domain detail panel configures the properties for external domain routing.

To view this administrative console page, click **Servers > Proxy Servers >** *server_name* **> SIP proxy settings > External Domains > New**

Domain

The SIP domain that is mapped to the protocol, host, and port that is specified in the fields on this panel. The SIP proxy server matches the domain that is found in the TO header of a SIP message to this value and uses the related information to connect to the specified SIP service.

Range	Valid SIP domain name, with the addition of an optional preceding * as a wildcard.
Setting recommendations	None

Protocol

The protocol that the SIP proxy server uses to connect to the SIP service.

Range	TCP, SSL, and UDP
Default	TCP
Setting recommendations	None

Distinguished name: The name that is associated with the external domain. Used when SSL client authentication is enabled to limit connections from an external domain.

Range	Any string
Default	blank
Setting recommendations	None

Host
The host that the SIP proxy server uses to connect to the SIP service.

Range	Valid host name or IP address
Default	blank
Setting recommendations	None

Port
The port that the SIP proxy server uses to connect to the SIP service.

Range	1 to 65535
Default	blank
Setting recommendations	None

SIP routing rules collection
Routing rules enable an administrator to direct SIP traffic to a specific cluster when there is more than one cluster running SIP applications in an WebSphere Application Server Network Deployment cell.

Routing rules are not needed in cases where there is one cluster running SIP. Rules are only applied to the initial message of a SIP conversation and not all SIP traffic. To view this administrative console page, click **Servers > Proxy Servers >** *server_name* > **SIP proxy settings > Routing rules.**

New
Clicking **New** launches the Routing Rule Detail Panel for creating a new rule.

Delete
Deletes selected rules.

Enable
Sets the enabled attribute of the selected rules to true.

Disable
Sets the enabled attribute of the selected rules to false.

Set order
Launches the Set Order panel.

Select
Allows the user to select multiple rows to be affected by the Delete, Enable and Disable actions.

Order
The order column represents the order in which the rules are evaluated. This is critical since a SIP message may match more than one rule.

Cluster
The name of the cluster to which the rule will route SIP traffic.

Condition
A concatenation of the list of conditions associated with the rule. Condition is not sorted.

Enabled/Disabled
Indicates whether the rule is enabled, and thus considered for evaluation.

SIP routing rules set order
It is possible that a SIP message can match more than one routing rule but the SIP proxy will stop evaluating at the first match. Routing rules are evaluated in the order that they appear in the configuration file. The set order routing rule panel enables the administrator to change this ordering.

Routing rules are not needed in cases where there is one cluster running SIP. Rules are only applied to the initial message of a SIP conversation and not all SIP traffic. To view this administrative console page, click **Servers > Proxy Servers >** *server_name* > **SIP proxy settings > Routing rules > Set Order.**

Move Up
Clicking **Move Up** moves the selected rule up one row.

Move Down
Clicking **Move Down** moves the selected rule down one row.

Select
Enables the user to select one row that will be acted on by the Move Up and Move Down actions.

Cluster
The name of the cluster that the rule will direct SIP requests to.

Condition
A concatenation of the list of conditions that are associated with the rule.

Enabled
Indicates whether the rule is enabled and thus considered for evaluation.

SIP routing rules detail
The routing rule detail panel provides the ability to create and modify individual rules. Since the structure of conditions is complex, the user will need to utilize a different set of panels to change the conditions associated with a rule.

Routing rules are not needed in cases where there is one cluster running SIP. Rules are only applied to the initial message of a SIP conversation and not all SIP traffic. To view this administrative console page, click **Servers > Proxy Servers >** *server_name* > **SIP proxy settings > Routing rules > New.**

Enabled
Enables a rule to be removed from consideration without requiring that it be deleted. This field is checked by default.

Data type	Boolean
Default	true
Setting recommendations	none
Setting dependencies	none

Target Cluster
The name of the cluster that the SIP message will be routed to if the message attributes match the conditions.

Data type	String
Default	First cluster in the list.

Range	List of existing clusters or “null cluster” to indicate that a request should be rejected.
Setting recommendations	None
Setting dependencies	None

SIP rule condition collection

Each rule contains a list of conditions that are combined using a logical AND operator. This means that all of the conditions need to be true for the rule to apply. This panel enables the user to manage the set of conditions.

To view this administrative console page, click **Servers > Proxy Servers >** *server_name* **> SIP Proxy Server Settings > Routing rules >** *target_cluster* **> Conditions**.

New

Create a new condition. Clicking **New** launches the Rule Condition Detail panel.

Delete

Removes a condition from the collection.

Type

Indicates which aspect of a SIP message the condition applies to.

Value

The value that will be compared to the aspect of the SIP message indicated by type.

SIP rule condition detail

This panel is used to create or modify a rule. There is one condition type that has a specific set of values, method, while the rest of the condition types are free form text. You can select between fixed method condition selection and free form entry for the other condition types.

Routing rules are not needed in cases where there is one cluster running SIP. Rules are only applied to the initial message of a SIP conversation and not all SIP traffic. To view this administrative console page, click **Servers > Proxy Servers >** *server_name* **> SIP Proxy Server Settings > Routing rules >** *target_cluster* **> Conditions > New**.

Condition type

Selects between creating a method condition with a fixed list of values and generic SIP message condition with an arbitrary set of values.

Default	Selected
Setting recommendations	None
Setting dependencies	If this button is selected, the associated drop-down list is enabled. The other entry controls are disabled.

Condition value

Contains the predefined method types used when the SIP method radio button is selected.

Range	INVITE, REGISTER, REFER, SUBSCRIBE, UNSUBSCRIBE, PUBLISH, MESSAGE, OPTIONS, INFO
Default	Invite
Setting recommendations	None
Setting dependencies	None

Condition type: Other
Contains the value against which the SIP message attribute is compared.

Default	Blank
Setting recommendations	None
Setting dependencies	None

SIP proxy inbound channel detail
This panel displays the configuration details of the SIP proxy inbound channel.

Routing rules are not needed in cases where there is one cluster running SIP. Rules are only applied to the initial message of a SIP conversation and not all SIP traffic. To view this administrative console page, click **Servers > Proxy Servers >** *server_name* > **Transport chain> SIP proxy inbound channel**.

Transport channel name
Specifies the name of the SIP proxy inbound channel.

The name field cannot contain the following characters: **# \ / , : ; " * ? < > | = + & %** '

This name must be unique across all channels in a WebSphere Application Server environment. For example, a SIP proxy inbound channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type	String

Discrimination weight
Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

Troubleshooting the proxy server
This topic helps you to solve problems that you might encounter with your proxy server.

Consult the following list if you are having problems with your proxy server:

- **The proxy server was created successfully, but I am unable to start it.** Check the SYSOUT file for port conflicts. Use the **netstat –a** command to see if any of the endpoints that are associated with the proxy server are already being used. You can find the ports in the administrative console by clicking **Servers > Proxy servers >** *<server_name>* **> Ports**.

  If the proxy server fails to start when attempting to start it as a non-privleged user on UNIX systems, check for the following message in the logs:

  ```
  ChannelFramew E   CHFW0029E: Error initializing chain HTTPS_PROXY_CHAIN because of
  exception
  com.ibm.wsspi.channel.framework.exception.RetryableChannelException: Permission
  denied
  TCPPort       E   TCPC0003E: TCP Channel TCP_7 initialization failed.  The socket
  bind failed for host * and port 80.  The port may already be in use.
  ```

  Change the ports of the PROXY_HTTP_ADDRESS and PROXY_HTTPS_ADDRESS transport chains to values greater than `1024`.

- **The proxy server started, but I am unable to access the application resources through the endpoints for the proxy server.** Ensure that the endpoints for the proxy server are among the host aliases in the virtual host that are associated with the application.
- **The proxy server routes to another core group.** Verify that core group bridges exist between the core groups in the cell, and that the processes that are chosen to be bridges are restarted. If there is a firewall between the core group, verify that the correct ports are open for core group bridge traffic.
- **The proxy server routes to another cell.** Review the core group bridge settings. Verify that the peer access point group names match in each cell. Check the peer ports against the bridge interfaces to verify that they are correct. If bridge interfaces or peer ports are added or changed, restart all bridge interfaces.
- **Receiving a blank page when making a request to the proxy.** Consider the following actions:
  - Update the virtual host. Ensure that the target application and routing rule are assigned to a virtual host that includes the proxy server listening ports (default: HTTP 80, HTTPS 443). Add the proxy server listening ports to the application, or routing rule virtual host, or use the proxy_host virtual host.
  - Stop the conflicting process. Check your system to ensure that no other process (for example, Apache, IBM HTTP Server, and so on) is running that uses the proxy server ports (default: HTTP 80, HTTPS 443). If this problem occurs, the proxy server seems to start normally, but is unable to receive requests on the affected listening port. Check your system as follows:
    1. Stop the proxy server.
    2. Query your system using **netstat** and **ps** commands to determine if an offending process is using the port on which the proxy server is listening.
    3. If an offending process is found, stop the process and configure your system so that the process is not started during system startup.
  - Enable proxy routing. Ensure that proxy routing is enabled for the Web module of the application. Proxy routing is enabled by default, so if no proxy properties are modified, disregard this solution. Otherwise, see “Customizing routing to applications” on page 432 for instructions on modifying the proxy properties.
  - Test direct request. Ensure that the target application is installed by making a request directly to the application server. If a response is not received, then the problem is with the application server and not the proxy server. Verify this case by going through the proxy server after you can receive a response directly from the application server.
- **HTTP 404 (File not found) error received from the proxy server**. Consider the following actions:
  - Update the virtual host. Ensure that the target application and routing rule are assigned to a virtual host that includes the proxy server listening ports (default: HTTP 80, HTTPS 443). Add the proxy server listening ports to the application, or routing rule virtual host, or use the proxy_host virtual host.
  - Enable proxy routing. Ensure that proxy routing is enabled for the Web module of the application. Proxy routing is enabled by default, so if no proxy properties are modified, disregard this solution. Otherwise, see “Customizing routing to applications” on page 432 for instructions on modifying the proxy properties.
  - Test direct request. Ensure that the target application is installed by making a request directly to the application server. If a response is not received, then the problem is with the application server and not the proxy server. Verify this case by going through the proxy server when you can receive a response directly from the application server.
- **Unable to make Secure Sockets Layer (SSL) requests to application or routing rule**. Ensure that the virtual host of the application or routing rule includes a host alias for the proxy server SSL port (default: 443).
- **Unable to connect to the proxy server...request times out**. Stop the conflicting process. Check your system to ensure that no other process (for example, Apache, IBM HTTP Server, and so on) is running that uses the proxy server ports (default: HTTP 80, HTTPS 443). If this situation occurs, the proxy server seems to start normally, but is unable to receive requests on the affected listening port. Check your system, as follows:
  1. Stop the proxy server.

2. Query your system using **netstat** and **ps** commands to determine if an offending process is using a port on which the proxy server is listening.
3. If an offending process is found, stop the process and configure your system so that the process is not started during system startup.

- **Did not receive a response from the error page application when the HTTP error occurred (for example, 404)**. Ensure that the error page URI is entered correctly. Also, make sure that the Handle remote errors option is selected if you are handling HTTP error responses from back-end servers. For more detailed information, refer to “Overview of the custom error page policy” on page 438 and the custom error page policy section of “Proxy server settings” on page 459.
- **What packages do I enable when tracing the proxy server?** All of the following packages are not needed for every trace, but if unsure, use all of them:
  - *=info
  - WebSphere Proxy=all
  - GenericBNF=all
  - HAManager=all
  - HTTPChannel=all
  - TCPChannel=all
  - WLM*=all
  - DCS=all
  - ChannelFrameworkService=all
  - com.ibm.ws.dwlm.*=all
  - com.ibm.ws.odc.*=all
- **How do I enable SSL on/off load?** SSL on/off load is referred to as the transport protocol in the administrative console, and transport protocol is a Web module property. Refer to “Customizing routing to applications” on page 432 to see how to configure Web module properties. No SSL on/off load or transport protocol properties exist for routing rules because the transport protocol is inherent to the generic server cluster that is specified in the routing rule.
- **When fronted by IBM HTTP Server or a plug-in, how do I configure the proxy server so I do not have to add a port for it to the virtual host?** For the proxy server to trust the security-related information, for example WebSphere Application Server private headers, of a request, add the originator of the request to the proxy server trusted security proxies list. For example, add an IBM HTTP Server or a plug-in sending requests to the proxy server to the proxy server trusted security proxies list. The plug-in sends WebSphere Application Server private header information that among other things, contains the virtual host information of a request. If the proxy does not trust the WebSphere Application Server private headers from the plug-in (or any client), the proxy server adds its own WebSphere Application Server private headers, which requires the addition of proxy server ports (HTTP and HTTPS) to the virtual host. Most likely, when using the plug-in with the proxy server, the intent is to use the proxy server as a back-end server. Be sure to add the WebSphere Application Server plug-in as a trusted security proxy to avoid having to expose the proxy server ports. Refer to “Routing requests from a plug-in to a proxy server” on page 436 for more information about configuring the WebSphere Application Server plug-in to use with the proxy server. Refer to “Proxy server settings” on page 459 for more information about trusted security proxies.
- **The proxy server seems to ″hang″ under stress, or ″Too Many Files Open″ exceptions display in ffdc or SystemErr.log.** Under high connection loads, the number of file system descriptors might become exhausted and the proxy server may seem to hang and drop ″Too Many Files Open″ exceptions in the `ffdc` directory or in the `SystemError.log` file because it is unable to open a socket. The problem can be alleviated by setting certain tuning parameters at the operating system level and at the proxy server level that optimize the use of connections for the proxy server:
  - **Operating system tuning for Windows 2000, 2003, and XP**
    - TcpTimedWaitDelay - Determines the time that must elapse before TCP/IP releases a closed connection and reuse its resources. This interval between closure and release is known as the

TIME_WAIT state, or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server costs less than establishing a new connection. By reducing the value of this entry, TCP/IP releases closed connections faster and can provide more resources for new connections. Adjust this parameter if the running application requires rapid release, the creation of new connections, or an adjustment because of a low throughput caused by multiple connections in the TIME_WAIT state.

View or set this value as follows:

1. Use the **regedit** command and access the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters` registry subkey to create a new REG_DWORD value named TcpTimedWaitDelay.
2. Set the value to decimal 30, which is Hex 0x0000001e. This value sets the wait time to 30 seconds.
3. Stop and restart your system.

Default value	0xF0, which sets the wait time to 240 seconds (4 minutes).
Recommended value	A minimum value of 0x1E, which sets the wait time to 30 seconds.

- MaxUserPort - Determines the highest port number that TCP/IP can assign when an application requests an available user port from the system. View or set this value as follows:
  1. Use the **regedit** command, access the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters registry` subkey, and create a new REG_DWORD value named MaxUserPort.
  2. Set this value to at least decimal 32768.
  3. Stop and restart your system.

Default value	None
Recommended value	At least decimal 32768.

– **Operating system tuning for Linux**

- timeout_timewait parameter - Determines the time that must elapse before TCP/IP releases a closed connection and can reuse its resources. This interval between closure and release is known as the TIME_WAIT state, or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server costs less than establishing a new connection. By reducing the value of this entry, TCP/IP can release closed connections faster, providing more resources for new connections. Adjust this parameter if the running application requires rapid release, the creation of new connections, and a low throughput due to many connections sitting in the TIME_WAIT state.

  View or set this value by issuing the following command to set the timeout_timewait parameter to 30 seconds:

  ```
  echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
  ```

- Linux file descriptors (ulimit) - Specifies the number of open files that are supported. The default setting is typically sufficient for most applications. If the value set for this parameter is too low, a file open error, memory allocation failure, or connection establishment error might display.

  View or set this value by checking the UNIX reference pages on the ulimit command for the syntax of different shells. Set the ulimit command to 65535 for the KornShell shell (ksh), by issuing the ulimit -n 65535 command. Use the ulimit -a command to display the current values for all limitations on system resources.

Default value	1024
Recommended value	65535

– **Operating system tuning for AIX**
  - TCP_TIMEWAIT - Determines the time that must elapse before TCP/IP releases a closed connection and can reuse its resources. This interval between closure and release is known as the TIME_WAIT state, or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server costs less than establishing a new connection. By reducing the value of this entry, TCP/IP can release closed connections faster, providing more resources for new connections. Adjust this parameter, if the running application requires rapid release or the creation of new connections, or if a low throughput occurs due to many connections sitting in the TIME_WAIT state.

    View or set this value by issuing the following command to set the TCP_TIMEWAIT state to 15 seconds:

    ```
    /usr/sbin/no -o tcp_timewait =1
    ```
  - AIX file descriptors (ulimit) - Specifies the number of open files that are permitted. The default setting is typically sufficient for most applications. If the value set for this parameter is too low, errors can occur when opening files or establishing connections, and a memory allocation error might display. To prevent WebSphere Application Server from running short on resources, remove the upper limits (ulimit) for resources on the user account on which the WebSphere Application Server process runs.

    View or set this value by changing the ulimit settings as follows:

    1. Open the command window.
    2. Type **smitty users** to open the AIX configuration program.
    3. Select **Change** or **Show Characteristics** of a user.
    4. Type the name of the user account on which the WebSphere Application Server runs.
    5. Press **Enter**.
    6. Change the following settings to the indicated value:

Soft FILE Size	-1
Soft CPU Time	-1
Soft STACK Size	-1
Soft CORE File Size	-1
Hard FILE Size	-1
Hard CPU Time	-1
Hard STACK Size	-1
Hard CORE File Size	-1

    7. Press **Enter** to save changes.
    8. Log out and log into your account.
    9. Restart WebSphere Application Server

Default value	2000
Recommended value	unlimited

– **Operating system tuning for Solaris**
  - TCP_TIME_WAIT_INTERVAL - Notifies TCP/IP on how long to keep the connection control blocks closed. After the applications complete the TCP/IP connection, the control blocks are kept for the specified time. When high connection rates occur, a large backlog of the TCP/IP connections accumulate and can slow server performance. The server can stall during certain peak periods. If the server stalls, the **netstat** command shows that many of the sockets that are opened to the HTTP server are in the CLOSE_WAIT or FIN_WAIT_2 state. Visible delays can

occur for up to four minutes, during which time the server does not send any responses, but CPU utilization stays high with all of the activities in system processes.

View or set this value by using the **get** command to determine the current interval and the set command to specify an interval of 30 seconds. For example:

```
ndd -get /dev/tcp tcp_time_wait_interval
ndd -set /dev/tcp tcp_time_wait_interval 30000
```

Default value	240000 milliseconds, which is equal to 4 minutes.
Recommended value	60000 milliseconds

– **Proxy server tuning**

- Persistent requests - A persistent request is one that is sent over an existing TCP connection. You can maximize performance by increasing the number of requests that are received over a TCP connection from a client. The value should represent the maximum number of embedded objects, for instance GIF and so on, in a Web page +1.

  View or set this value in the WebSphere Application Server administrative console by clicking **Servers > Proxy Servers >** *server_name* **> Proxy server transports > HTTP_PROXY_CHAIN/ HTTPS_PROXY_CHAIN**

Default value	100
Recommended value	A value that represents the maximum number of embedded objects in a Web page + 1.

- Outbound connection pool size - The proxy server pools outbound connections to target servers and the number of connections that resides in the pool is configurable. If the connection pool is depleted or empty, the proxy server creates a new connection to the target server. Under high concurrent loads, increase the connection pool size should to a value of the expected concurrent client load to achieve optimal performance.

  View or set this value in the WebSphere Application Server administrative console by clicking **Servers > Proxy Servers >** *server_name* **> HTTP Proxy Server Settings**. In the Content Server Connection section, increase the maximum connections per server field to a value that is equal to or greater than the expected maximum number of connected clients. Save your changes, synchronize the changes to the proxy server node, and restart the proxy server.

Recommended value	Value consistent to the expected concurrent client load.

- Outbound request time-out - Often times, the back-end application servers that are fronted by the proxy server may be under high load and may not respond in an adequate amount of time, therefore the connections on the proxy server may be tied up from waiting for the back-end application server to respond. Alleviate this by configuring the amount of time the proxy server waits for a response from the target server. This is the Outbound Request Time-out value. By managing the amount of time the proxy server waits for a slow back-end application server, connections are freed up faster and used for other request work.

  View or set this value in the WebSphere Application Server administrative console by clicking **Servers > Proxy Servers** *server_name* **> HTTP Proxy Server Settings**. In the Content Server Connection section, set Outbound Request Time-out to a value that represents the acceptable response time from the point of view of the client.

Default value	120
Recommended value	A value that represents the acceptable response time from the point of view of the client.

Troubleshooting request routing and workload management through the proxy server

This section provides information for how to troubleshoot request traffic that flows through the proxy server.

You will need to know the machines and nodes that will belong to the proxy server cluster. WebSphere Application Server V6.1 needs to be installed on those machines. You will also need to know the URL for the applications, application deployment, and cluster definition details. The proxy server should be started.

You can use the proxy server MBean to determine how requests are routed to applications, and subsequently, to a particular application server. If the request is being routed incorrectly, you can disable routing to specific applications or reconfigure the routing rules.

1. Obtain the Dynamic Route MBean for the proxy server and invoke the operation to generate routing information for the URI. Start **wsadmin** and get all of the Dynamic Route MBeans as follows:

   ```
   $AdminControl queryNames
   type=DynamicRoute,*

   set routembean <cut and paste the MBean Identifier from the previous command output>

   $AdminControl invoke $routembean debugRouting {http://*/urlpattern all}
   ```

   Use an asterisk (*) to match all of the virtual hosts, or explicitly specify a virtual host. For example, `http://proxy_name:80/urlpattern`. The **set routembean** command should correspond to the MBean from the output of the previous command.

   The proxy server will start generating routing-related information for all subsequent HTTP requests that match the specified virtual host and URL pattern to the `SystemOut.log` file.
2. Send representative workload traffic through the proxy server.
3. Analyze the routing information in the proxy server `SystemOut.log` file.
4. Make required changes to application routing to enable or disable routing through the proxy server, using the administrative console, by clicking **Applications > Enteprise Applications**.
5. Repeat steps two through four until the routing of all requests are satisfied.
6. Disable gathering routing information using **wsadmin** as follows:

   ```
   $AdminControl invoke $routembean
   stopDebugRouting
   ```

The proxy server and the applications are correctly configured for external access.

Proxy server collection

This topic lists the proxy servers in the cell. A proxy server resides within a WebSphere Application Server Network Deployment node.

A proxy server is used to classify, prioritize, and route HTTP and SIP requests to servers in the enterprise as well as cache content from servers. You can use this page to create, delete, or modify a proxy server.

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

To configure the proxy server to route work to V6 WebSphere Application Servers in another cell, use core group bridge settings (**Servers > Core groups > Core group bridge settings**), which sets up communication between cells.

Currently, configuring the proxy server to route work to a V6 WebSphere Application Server - Express cell requires advanced configuration.

To configure the proxy server to route work to an application server which is not an IBM WebSphere Application Server or a pre-V6 WebSphere Application Server cell, the following advanced configuration is required:

1. Define a generic server cluster. From the administrative console, click **Servers > Generic Server Clusters**.
2. Define a URI group. From the administrative console, click **Environment > URI Groups**.
3. Create routing rules. From the administrative console, click **Servers > Proxy Servers >** *server_name* **> Proxy Server Properties > Routing rules**.

Both generic server clusters and URI groups are also accessible in the administrative console under Related Items for the proxy server.

Name

Specifies a logical name for the proxy server. For WebSphere Application Server, 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 Application Server uses the server name for administrative actions, such as referencing the server in scripting.

Node

The name of the node where the proxy server resides.

Version

Indicates the WebSphere Application Server version you are running.

Protocol

Indicates the protocol or protocols that the proxy server is configured to handle. This information is based on the types of transport channels that are included in the transport chains that are configured for the proxy server.

For example, if a transport chain includes an HTTP channel, `HTTP` displays in this field. If a transport chain includes both a SIP and an HTTP channel, `SIP,HTTP` displays in this field.

Status

Indicates whether the proxy server is started, stopped, or unavailable.

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.

Proxy server configuration

You can modify an existing proxy server to perform advanced routing options, such as routing requests to a non-WebSphere Application Server cell, and to perform caching. The options to configure the proxy server from this panel are under Proxy server properties.

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

On the **Configuration** tab, you can edit proxy server setting fields.

Name

Indicates a logical name for the proxy server. Proxy server names must be unique within a node.

Run in development mode

Enabling this option can reduce the startup time of a proxy server. This time can include Java Virtual Machine (JVM) settings, such as disabling bytecode verification and reducing just-in-time (JIT) compilation costs. Do not enable this setting on production servers.

Specify this option if you want to use the -Xverify and -Xquickstart JVM settings on startup. After selecting this option, save the configuration and restart the proxy server to activate development mode.

The default setting for this option is `false`, which indicates that the proxy server is not started in development mode. Setting this option to `true` specifies that the proxy server is started in development mode with settings that speed server startup time.

Data type	Boolean
Default	false

Parallel start

Select this field to start the proxy server on multiple threads. This option might shorten the startup time.

Specify this option if you want the proxy server components, services, and applications to start in parallel, rather than sequentially.

The default setting for this option is `true`, which indicates that the proxy server is started using multiple threads. Setting this option to `false` specifies that the server does not start using multiple threads which might lengthen startup time.

The order in which the applications start depends on the weights that you assigned to each of them. Applications that have the same weight are started in parallel. You set the weight of an application with the **Starting weight** option on the **Applications > Enterprise Applications >** *application_name* page of the administrative console.

Data type	Boolean
Default	true

Proxy server settings

Use this topic to perform advanced configuration on a proxy server. Proxy settings enable the system administrator to fine tune the behavior of the proxy server. In particular, you can configure the connections and requests to the application server, enable caching, configure the requests that must be rejected, define how error responses are handled, and specify the location of the proxy logs.

The proxy server, upon creation, auto-senses the environment and is capable of routing requests to WebSphere Application Server. Additional configuration can be applied to the proxy server to meet the needs of a particular environment.

To view this administrative console page, click **Servers > Proxy Servers >** *server_name* **> HTTP Proxy Server Settings > Proxy settings**.

You can edit configurable field settings for the proxy server on the Configuration tab.

Content server connection

Configure basic HTTP connection parameters between the proxy server and content servers.

Outbound request timeout: The default number of seconds the proxy server waits for a response before timing out a request to a content server. Consider this option carefully when changing the value.

Outbound connect timeout: The number of milliseconds that the proxy server waits to connect to a server. If this time expires, the proxy server attempts to connect to a different server. If no other available servers exist, the request times out. A value of 0 indicates that the proxy server should use the operating system kernel timeout value.

Pool connections to content server: The option to pool connections to the server is an optimization feature. Pooling prevents the need to frequently create and destroy socket connections to the server, by allowing the proxy server to pool these connections and reuse them.

Maximum connections per server: The maximum number of connections that will be pooled to any single content server. Proxy custom properties that tweak content server connections are as follows:

- key=http.maxTargetReconnects: Maximum number of reconnects to the same target content server for each request. The default is 5.
- key=http.maxTargetRetries: Maximum number of times the proxy will attempt to select a new target content server for each request. The default is 5.
- key=http.routing.sendReverseProxyNameInHost: Determines whether or not the proxy server name is placed in the host header for content that is not specific to WebSphere Application Server content servers. The options are `true` or `false` and are not case sensitive. The default is false.
- key=http.compliance.disable: Determines whether HTTP V1.1 compliance is enforced on proxy content server connections. The options are `true` or `false` and are not case sensitive. The default is false.
- key=http.compliance.via: The value of the via header that is appended to requests and responses for HTTP compliance. If the value is null, a via header will not be appended. If the value is true, a default via value is appended. Otherwise, the specified string via value is appended. The default is null.

SSL Configuration: Set the SSL configuration from one of several sources:

Centrally managed	Use the SSL configuration that is scoped for this endpoint.
Specific to this endpoint	Use a specific SSL configuration.
Select SSL Configuration	Options are NONE, CellDefaultSSLSettings, AdminSoapSSLSettings, and NodeDefaultSSLSettings

Caching

The proxy server can be configured to cache the content of servers.

By default, caching content is enabled. The properties that follow apply only if caching is enabled:

- **Enable caching**: Enables caching framework for the proxy server and enables static content caching, as defined by HTTP 1.1 specifications.
- **Cache instance name**: The dynamic cache object cache instance, that is configured in detail under **Resources > Cache instances > Object cache instances**, used to cache all static and dynamic content responses. This object cache instance must be configured to support new I/O (NIO) application program interfaces (APIs).
- **Cache SSL content**: Determines whether client proxy server SSL connections that are terminated by the proxy server should have their responses cached.
- **Cache aggressively**: Enables caching of HTTP responses that would not normally be cached. Caching rules that are defined by HTTP 1.1 may be broken in order to gain caching optimizations.

- **Cache dynamic content**: Determines whether dynamic content that is generated by WebSphere Application Servers V6.02 or later is cached. Caching dynamic content generated by content servers prior to WebSphere Application Server V6.02 is not supported.

Enable Web services support

Check this option to enable the proxy server to route Web services traffic.

Exclusions

The proxy server examines every incoming request. You can define certain methods for exclusion and if the requested HTTP method matches any of the configured methods for exclusion, the proxy server rejects the requests with a METHOD DISALLOWED error.

Logging

The proxy server has logs that are generated for proxied requests and stored cache requests. With this configuration, you can specify the location of the proxy access log and the cache access log.

Use the default location, or specify a directory location. There is a third log called `${SERVER_LOG_ROOT}/local.log` that logs locally-served proxy content. This content does not come from the proxy cache.

Proxy custom properties that can be used to tweak logging are as follows:

- key=http.log.disableAll: This property disables all logging. A value of `true` stops proxy, cache, and local logging.
- key=http.log.maxSize:The maximum log size in megabytes (MB). A value of `UNLIMITED` indicates unlimited. 25 MB is the default.
- key=http.log.localFileName: Contains the name of the local log. A value of `NULL` indicates that the default `${SERVER_LOG_ROOT}/local.log` is used.

HTTP requests are logged in one of three logs: proxy, cache, and local. Local log configuration is not currently available in the administrative console, but it is available at ${SERVER_LOG_ROOT}/local.log. Specify the location of this log by setting the http.log.localFileName custom property to the file location. The content of each log is formatted using National Center for Supercomputing Applications (NCSA) common log format.

- Proxy access log: Logs responses that are received from remote servers.
- Cache access log: Logs responses that are served from the local cache.
- Local access log: Logs all non-cache local responses, for example, redirects and internal errors.

Security

Use this section to set up security options.

- **Trusted security proxies**: Topologies exist where another layer of work routing is enabled on top of the proxy server. For example, Web servers read incoming requests to verify which proxy they are routed to. This configuration field enables intermediaries other than the proxy server to handle the request by explicitly telling the proxy server that is to trust them. Use an IP or fully qualified host name in this field.

  An empty list of trusted security proxies indicates that all WebSphere Application Server plug-in clients are trusted. Once a trusted security proxy is specified, only the listed clients will be trusted.
- **Server header**: Enables configuration of the HTTP server header that is returned to clients. Used to suppress server information. If the value is `""`, the content server name is forwarded to client. If the value is `"TRUE"`, the default server name `"WebSphere Proxy"`, is sent as the content server name. If the value is anything else, the value that is specified is sent as the content server name.

Proxy plugin configuration policy

- **Generate plugin configuration**: Use this parameter for the generation of a proxy plugin configuration file that you can use on a Web server that is deployed in front of the proxy server. The plugin can determine the URI that the proxy is handling on behalf of the application server. The plugin can determine the endpoint, or boundaries of the proxy so that it can properly route requests that it receives to the proxy. This feature is useful for those who prefer to deploy a proven Web server in the demilitarized zone (DMZ), which is fully capable of exploiting the ability of the proxy server.

  Options are available to define a level by which to generate the plugin, as follows:

Scope	Description
None	No scope.
All	The proxy server generates a plugin configuration that includes all of the URIs that are handled by proxy servers in the local cell and all cells that are connected by a core group bridge.
Cell	The proxy server generates a plugin configuration that includes all of the URIs that are handled by all the proxy servers in the cell.
Node	Includes all of the URIs that are configured for the node.
Server	The proxy server generates a plugin configuration file only for the proxy server that is currently configured.

- **Plugin config change script**: Specifies the path to a script that is run after the WebSphere Application Server plugin configuration is generated.

Custom error page policy

Use this field to support the use of customized error pages when errors occur during the processing of the request.

The default is no customized error pages generated. The properties that follow enable customized error pages for use when errors occur during request processing:

- **Error page generation application URI**: If a valid URI to an installed application is not provided, the custom error page policy does not handle requests.
- **Handle remote errors**: When not selected, only HTTP response error status codes generated by the proxy server are handled. When selected, HTTP response error status codes generated by the proxy server and HTTP response error status codes generated elsewhere after the proxy on the proxy content server connection error responses are handled. A best practice is to configure an error page application on the same physical machine as the proxy server.
- **Headers to forward to error page application**: Specifies additional header values from the client request to forward to the error page application as query parameters. The responseCode and URI query parameters are always sent to the error page application, in addition to the ones that are configured. The responseCode parameter is the HTTP status code that generates internally or is returned by the content server. The URI parameter is the request URI for the client.

  Example - The error page URI is `/ErrorPageApp/ErrorPage`, the headers to forward contain `Host`, and a client sends the following request:

  ```
  GET  /house/rooms/kitchen.jpg HTTP/1.1
  Host:  homeserver.companyx.com
  ```

  The request results in a HTTP 404 response (local or remote), and the request URI to the error page application would be:

  ```
  /ErrorPageApp/ErrorPage?responseCode=404&uri=/house/rooms/kitchen.jpg&Host= homeserver.companyx.com
  ```

- **HTTP status codes that are to be recognized as errors**: The status codes that the error page policy provide a response for. If a status code is not specified, the original content of responses with that

status code are returned. If no HTTP status codes are specified, the defaults, 404 and 5XX, are used. Instead of specifying status codes individually, the following method is recommended to represent a range:

- 5XX: 500-599
- 4XX: 400-499
- 3XX: 300-399
- 2XX: 200-299

Proxy custom property to use when tweaking the custom error page: key=http.statuscode.errorPageRedirect. This custom property determines whether error page generation is done using the redirect, instead of using the proxy error page application. The values are true or false. The default is false.

Generic server clusters collection

Use this page to create, delete or modify a generic server cluster. Creating a generic server cluster is the next step towards generating the ability to route requests to a non-IBM WebSphere Application Server or a pre-V6 WebSphere Application Server cell, after creating the proxy server.

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

The system administrator can use the Generic Server Cluster panel to configure external servers, which are non-IBM WebSphere Application Server or a pre-V6 WebSphere Application Server, to create a logical cluster the proxy server can route work to. A generic server cluster defines the server endpoints that URI groups that are mapped to.

After you have created a generic server cluster, you want to create cluster members and define the cluster endpoints. Select the generic server cluster you created and click **Ports**.

Name

Specifies a logical name for the cluster. This is a user-defined field.

The name field cannot contain the following characters: **# \ / , : ; " * ? < > | = + & %** '

The name that defined must be unique among generic server clusters and cannot begin with a period or a space. A space does not generate an error, but leading and trailing spaces are automatically deleted.

Filter

Select the column by which to filter.

Search term(s)

Specifies the filter criteria. This is a user-defined field.

Generic server clusters configuration

Use this topic to configure a generic server cluster. Creating a generic server cluster is the next step, after creating the proxy server, towards generating the ability to route requests to a non-IBM WebSphere Application Server or a pre-V6 WebSphere Application Server cell.

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

Now that you have configured a generic server cluster, you can create cluster members and define the cluster endpoints. Click **Additional properties** > **Ports**.

You can edit generic server cluster configurable field settings on the Configuration tab.

Name
Specifies the user-defined name for the cluster.

Protocol
The protocol field determines whether or not secure communication is used when connecting to members of the cluster.

The choices are HTTP and HTTPS.

Generic server cluster ports collection

After defining the generic server cluster name, use this page to create, delete, and configure the members of the cluster.

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

Host
The host name is either an IP address or the qualified host name of the cluster member.

Specify a valid host name.

Generic server cluster members

After defining the generic server cluster name, use this page to define the members of the cluster.

To view this administrative console page, click **Servers** > **Generic Server Clusters** > *server_name* > **ports** > *host_name*.

After you complete this task, you can create a URI group and then define routing rules.

You can edit generic server cluster member field settings on the Configuration tab.

Host
The host name is either an IP address or the qualified host name of the cluster member.

Specify a valid host name.

Port
Enter the port on which the host name is listening. This entry ensures that the proxy server can communicate with the cluster member.

Specify a valid port number.

Weight
The load balancing weight is used to determine how frequently the cluster member is routed, relative to the other members in the group. The higher the weight the more frequently the cluster member is routed to.

The recommended weight value is between 1 and 20. A value of 0 will result in a cluster member that will never be routed to.

Routing rules

Use this topic to set the advanced configuration routing rules to ensure work requests arrive at the proper generic server cluster. From this topic you can create, delete, or modify a routing rule.

To view this administrative console page, click **Servers > Proxy Servers >** *server_name* > **HTTP Proxy Server Settings > Routing Rules.**

Before you create routing rules, which are used to route requests to servers, you must define a generic server cluster (**Server > Generic Server Clusters**), a URI group (**Environment > URI Groups**), and optionally appropriate virtual hosts (**Environment > Virtual hosts > default host**).

Routing rules are used to assist the routing of work requests to non-IBM WebSphere Application Server nodes. In addition, using routing rules, a system administrator can reroute work without heavily impacting the environment. This capability is useful when nodes are taken down for maintenance.

For example, the system administrator can set up a routing rule to route /images/* to the ImageServerCluster generic server cluster. If the **ImageServerCluster** cluster has to come down, the administrator can then route /images/* to another cluster with similar capability, or use a redirect rule. This situation explains why the URI group can be defined independently of the generic server cluster. If the generic server cluster must come down, the URI group can be rerouted elsewhere. When you create the generic server cluster by providing a name, you can configure the cluster by using the ports link to create the actual cluster members.

Routing rules function by using the configured virtual hosts and URIs as matching criteria. The proxy server scans all incoming requests and compares the URI and host header from it and matches it against the virtual host and URIs that are configured in the rule. You must create the URI group for a routing rule before creating the routing rule. If you are routing to a generic server cluster, you must also create the cluster before defining the routing rule. You can create the URI group by completing the following tasks:

1. Create the routing rule name.
2. Determine if you want to enable this rule. You can create routing rules and not enable them. This capability is useful when planning for the maintenance of nodes or for emergency planning.
3. Select the virtual host name from the drop-down menu. The virtual host name field is a selectable field that is preconfigured with the defined virtual hosts in the cell. If you do not see the virtual host that you want in the menu, click **Environment > Virtual Hosts** and define the host there.
4. Select the URI group for the routing rule. The URI group field is populated with all the preconfigured URI groups in the cell. If you do not see the URI group that you are looking for, click **Environment > URI Groups** and create one.
5. Select and define a routing rule. This option specifies how to route a request that matches the defined virtual host and URI group. The three options for this field are:
   - Generic Server Cluster: Routes requests to a preconfigured generic server cluster. Use the drop-down box to select the generic server cluster.
   - Fail: Rejects requests by returning the specified HTTP status code.
   - Redirect: Redirects a client to the specified URL. This option can be used to ensure a request is routed through Secure Sockets Layer (SSL).

Name

The name field is required and is a user-defined field.

The name field cannot contain the following characters: **# \ / , : ; " * ? < > | = + & %** '

The name that is defined must be unique among routing rules and cannot begin with a period or a space. A space does not generate an error, but leading and trailing spaces are automatically deleted.

Routing rules configuration

Use this topic to create the advanced configuration routing rules to ensure that work requests arrive at the proper generic server cluster.

To view this administrative console page, click **Servers > Proxy Servers >** *server_name* **> Proxy Server Properties > Routing Rules >** *rule_name*.

You can edit routing rules configurable field settings on the Configuration tab.

Name

The name field is required and is a user-defined field.

The name field cannot contain the following characters: `# \ / , : ; " * ? < > | = + & % '`

The name that is defined must be unique among routing rules and cannot begin with a period or a space. A space does not generate an error, but leading and trailing spaces are automatically deleted.

Enable this rule

You can create routing rules and then not enable them. This capability is useful when planning for the maintenance of nodes or emergencies.

By default, the rule is enabled.

Virtual host name

The virtual host name field is a selectable field that is preconfigured with the defined virtual hosts in the cell.

If you do not see the virtual host that you are looking for in the menu, click **Environment > Virtual Hosts** from the administrative console and define the virtual host there.

URI group

The URI group field is populated with all the preconfigured URI groups in the cell.

If you do not see the URI group that you are looking for, click **Environment > URI Groups** and create one.

Routing rule

This option specifies to the proxy server how to route a request that matches the given criteria (virtual host and URI group) that you defined. You have three options for this field.

Generic Server Cluster: If you want only the proxy server to seek a preconfigured generic server cluster, select that option and use the drop-down box to select the generic server cluster.

Failure Status Code: If you want to reject the requests that match the specific criteria, you use the failure status code and provide an HTTP status code to use in the response to the sender.

Redirect URL: Use this last option to send a redirect to the client. If you select this option, enter a fully qualified URL like `http://abc.xyz.com`. Usually, the URL is somewhere within the enterprise, sometimes right back to the proxy on a different port. You can use this option to ensure a request is routed through protocols like Secure Sockets Layer (SSL).

URI groups

A group of URI patterns, which you define, that can be mapped back to generic server clusters. When creating a URI group, verify that you are planning for URIs that form a logical collection. From this topic, you can create, delete, or modify a URI group.

To view this administrative console page, click **Environment > URI Groups**.

Name

The URI group name is a user-specified name.

The name field cannot contain the following characters: **# \ / , : ; " * ? < > | = + & %** '

The name that is defined must be unique among URI groups and cannot begin with a period or a space. A space does not generate an error, but leading and trailing spaces are automatically deleted.

URI group configuration

Use this topic to configure a URI group that can be mapped back to generic server clusters. When creating a URI group, ensure that you are planning for URIs that form a logical collection.

After you create a generic server cluster and a URI group, you are ready to define the routing rules that map the URI group to the generic server cluster. The routing rules ensure that the requests for specific URIs go to the proper generic server cluster.

To view this administrative console page, click **Environment > URI Groups >** *URI_group_name*.

You can edit URI groups configuration fields on the Configuration tab.

Name

The name field is required and is a user-defined field.

The name field cannot contain the following characters: **# \ / , : ; " * ? < > | = + & %** '

The name that is defined must be unique among URI groups and cannot begin with a period or a space. A space does not generate an error, but leading and trailing spaces are automatically deleted.

URI pattern

Use this field to define URI patterns that constitute the URI group.

The patterns can be any valid URI and can contain one or more wildcard characters.

Rewriting rules collection

Use this page to define how to rewrite URLs in a request or response.

To view this administrative console page, click **Servers > Proxy servers >** *server_name* **> HTTP proxy server settings > Rewriting rules**.

From URL pattern

Specifies the original URL pattern in the 302 response header from the target server.

To URL pattern

Specifies how that URL should be modified so that the client is redirected to the proxy server or an appropriate public URL.

Rewriting rules configuration

Use this topic to rewrite redirected URLs.

Rewriting rules define how the proxy server will rewrite URLs. The proxy server currently only supports rewriting redirected responses. Responses that have been redirected by target servers typically return a 302 status code with a location header that defines the URL that the client should be redirected to. Rewriting this URL is necessary if the target server is not aware of the proxy servers. The redirected URL is modified to correctly point clients to the proxy server instead of directly to a target server that may not be visible to clients.

To view this administrative console page, click **Servers > Proxy servers >** *server_name* **> HTTP proxy server settings > Rewriting rules > New**.

From URL Pattern

Specifies the original URL pattern in the 302 response header from the target server.

The pattern can include the following wild card symbol: *****

To URL Pattern

Specifies how that URL should be modified so that the client is redirected to the proxy server or an appropriate public URL.

The pattern can include the following wild card symbol: *****

A rule with a from URL pattern of `http://internalserver/*` and to url pattern of `http://publicserver/*` would cause a redirected response with the original location header of `http://internalserver/secure/page.html` to be rewritten as `http://publicserver/secure/page.html`.

Static cache rules collection

This topic lists the static cache rules for a proxy server. From this topic you can create, delete, or modify a static cache rule.

To view this administrative console topic, click **Servers > Proxy Servers >** *server_name* **> HTTP Proxy Server Settings > Static cache rules**.

URI Groups

The URI group name is a user-specified name.

The name field cannot contain the following characters: **# \ / , : ; ″ * ? < > | = + & % ’**

The name that is defined must be unique among URI groups and cannot begin with a period or a space. A space does not generate an error, but leading and trailing spaces are automatically deleted.

Disable caching for this URI group

Specifies whether or not caching is disabled.

The default is `false`, which indicates that caching is enabled for the URI group.

Default expiration

The default expiration that you set for the cached response for the URI that is associated with this cache rule.

The default expiration value is in seconds.

Last modified factor

Use this field to derive the cache expiration value for a response if it does not have HTTP expiration headers and when it has a LastModifiedTime header in the response.

Name of the virtual host

A virtual host that is configured using the virtual host service. This virtual host is associated with the proxy server. This attribute is one of the elements in a request that is matched by the proxy server to determine if this rule is activated.

Static cache rule settings

Use this topic to configure a cache rule that is associated to a URI group for the proxy server. HTTP 1.1 defines a set of rules for proxy servers to cache content. Static cache rules enable these default rules to be overridden for a given address space. In order for the rules to have meaning, you must enable caching on the **Servers** > **Proxy Servers** > *server_name* > **HTTP Proxy Server Settings** > **Proxy settings** administrative console page.

To view this administrative console page, click **Servers** > **Proxy Servers** > *server_name* > **HTTP Proxy Server Settings** > **Static cache rules** > *URI_group*.

You can edit proxy server setting fields on the Configuration tab.

URI groups

URI groups, along with the virtual host, define the scope of the address space to have cache customizations performed.

The name field cannot contain the following characters: **# \ / , : ;** ″ *** ? < > | = + &** % ’

The name that is defined must be unique among URI groups and cannot begin with a period or a space. A space does not generate an error, but leading and trailing spaces are automatically deleted.

Disable caching for this URI group

Disables caching for this address space. A user may wish to disable caching for a set of content servers that are known to contain sensitive or highly-personalized information.

The default is `false`, which indicates that caching is enabled for the URI group.

Default expiration

The default expiration value, in seconds, that is used to determine the validity of a cached response when all other HTTP 1.1 caching-related response headers do not give guidance. The default value is sufficient in most environments.

The default expiration value is in seconds.

Last modified factor

The percentage of a last-modified header for a response that determines the validity of a cached response when the response does not have explicit HTTP expiration headers. The default value is sufficient in most environments.

The default for this field is `0.0`.

Name of the virtual host

A virtual host that is configured using the virtual host service. This virtual host is associated with the proxy server. This attribute is one of the elements in a request that is matched by the proxy server to determine if this rule is activated.

The default for this field is `none`.

HTTP proxy inbound channel settings

Use this page to view and configure an HTTP proxy inbound channel. This type of transport channel provides the HTTP proxy capabilities.

To view this administrative console page, click **Servers > Proxy Servers >** *server_name* **>** **Proxy settings> Proxy server transports > HTTP_PROXY_CHAIN > ProxyInboundChannel (PROXY_1)**.

Transport channel name

Specifies the name of the HTTP proxy inbound channel.

The name field cannot contain the following characters: **# \ / , : ; " * ? < > | = + & % '**

This name must be unique across all channels in a WebSphere Application Server environment. For example, an HTTP proxy inbound channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type	String

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

Appendix. Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories infer specific default directory locations. This topic describes the conventions in use for WebSphere Application Server Network Deployment.

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server Network Deployment products or components, of course, require multiple locations.

app_server_root - the install_root for WebSphere Application Server
The default installation root directory for WebSphere Application Server Network Deployment is the /QIBM/ProdData/WebSphere/AppServer/V61/ND directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server Network Deployment is the /QIBM/UserData/WebSphere/AppServer/V61/ND/profiles/*profile_name* directory.

app_server_user_data_root - the user_data_root for WebSphere Application Server
The default user data directory for WebSphere Application Server Network Deployment is the /QIBM/UserData/WebSphere/AppServer/V61/ND directory.

plugins_root
The default installation root directory for Web server plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V61/webserver directory.

plugins_user_data_root
The default Web server plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V61/webserver directory.

plugins_profile_root
The default Web server plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V61/webserver/profiles/*profile_name* directory.

app_client_root
The default installation root directory for the J2EE WebSphere Application Client is the /QIBM/ProdData/WebSphere/AppClient/V61/client directory.

app_client_user_data_root
The default J2EE WebSphere Application Client user data root is the /QIBM/UserData/WebSphere/AppClient/V61/client directory.

app_client_profile_root
The default J2EE WebSphere Application Client profile root is the /QIBM/UserData/WebSphere/AppClient/V61/client/profiles/*profile_name* directory.

web_server_root
The default web server path is /www/*web_server_name*.

shared_product_library
The shared product library, which contains all of the objects shared by all Version 6.1 installations on the system, is QWAS61. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

product_library
The product library, which contains program and service program objects (similar to .exe, .dll, .so objects for Windows, Linux, and UNIX operating system platforms), is: QWAS61*x* where *x* is A, B, C, ... For the default installation, the value is QWAS61A.

product_lib

The product library that contains the service program objects for the web server plugins. For the default Web Server Plugins install, this is QWAS61A. If you install the Web Server Plugins multiple times, the product_lib is QWAS61c, where *c* is B, C, D, ... The plugins_install_root/properties/product.properties contains the value for the product library..

cip_app_server_root

The default installation root directory is the /QIBM/ProdData/WebSphere/AppServer/V61/ND/cip/*cip_uid* directory for a customized installation package (CIP) produced by the Installation Factory.

A CIP is a WebSphere Application Server Network Deployment product bundled with optional maintenance packages, an optional configuration archive, one or more optional enterprise archive files, and other optional files and scripts.

cip_user_data_root

The default user data root directory is the /QIBM/UserData/WebSphere/AppServer/V61/ND/cip/*cip_uid* directory for a customized installation package (CIP) produced by the Installation Factory.

cip_profile_root

The default profile root directory is the /QIBM/UserData/WebSphere/AppServer/V61/ND/cip/*cip_uid*/profiles/*profile_name* directory for a customized installation package (CIP) produced by the Installation Factory.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Intellectual Property & Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Trademarks and service marks

For trademark attribution, visit the IBM Terms of Use Web site (http://www.ibm.com/legal/us/).