

IBM WebSphere Application Server for IBM i, Version 8.0

*Setting up the application serving
environment*

IBM

Note

Before using this information, be sure to read the general information under “Notices” on page 255.

Compilation date: July 14, 2011

© Copyright IBM Corporation 2011.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

How to send your comments	vii
Changes to serve you more quickly	ix
Chapter 1. Configuring port settings	1
Updating ports in existing profiles	2
Chapter 2. Managing profiles on non-z/OS operating systems	3
Profile concepts	3
Profiles: File-system requirements	6
Managing profiles using commands	7
manageprofiles command	8
Managing default profiles	20
Default application server profiles	20
Default secure proxy profiles	21
Default application client profiles	22
Default remote HTTP profiles	22
Deleting profiles	22
Chapter 3. Administering nodes and resources	25
Working with nodes - groups of managed servers	25
Changing host names	25
Administering stand-alone nodes using the administrative agent	27
Administrative agent	28
Administrative agent security	29
Setting up the administrative agent environment	30
Starting and stopping the administrative agent	33
Administrative agent settings	35
Node collection for the administrative agent	36
Unregistering nodes of the administrative agent	36
Configuring administration services	38
Remote files services for file transfer and file synchronization	38
Repository service settings	39
Repository service settings	39
Java Management Extensions connector properties	39
Java Management Extensions (JMX) connectors	47
SOAP connector and Inter-Process Communications connector properties files	48
Extension MBean Providers collection	49
Extension MBean collection	50
Administrative audit messages in system logs	51
Java Management Extensions connector properties	52
Java Management Extensions (JMX) connectors	59
SOAP connector and Inter-Process Communications connector properties files	61
Extension MBean Providers collection	61
Extension MBean collection	62
Administrative audit messages in system logs	63
Administration service settings	64
Remote connector	64
Local connector	64
Administration services custom properties	65
com.ibm.websphere.mbeans.disableRouting	65
Administrative topology: Resources for learning	65

Chapter 4. Working with server configuration files	67
Configuration documents	67
Configuration document descriptions	69
Object names: What the name string cannot contain	71
Handling temporary configuration files resulting from session timeout	72
Changing the location of temporary configuration files	73
Changing the location of backed-up configuration files	73
Changing the location of the wstemp temporary workspace directory	73
Backing up and restoring administrative configuration files	75
Backing up and recovering administrative configurations	75
Server configuration files: Resources for learning	76
Configuration problem settings	77
Configuration document validation	77
Enable Cross Validation	77
Configuration Problems	77
Scope	77
Message	77
Explanation	77
User action	78
Target Object	78
Severity	78
Local URI	78
Full URI	78
Validator classname	78
Runtime events	78
Message details	79
Chapter 5. Administering application servers	81
Configuring virtual hosts	82
Virtual hosts	83
Virtual host collection	86
Creating, editing, and deleting WebSphere variables	89
WebSphere variables collection	91
Introduction: Variables	92
WebSphere variables	93
Configuring the IBM Toolbox for Java	95
Managing shared libraries	96
Creating shared libraries	98
Shared library collection	100
Associating shared libraries with applications or modules	103
Associating shared libraries with servers	105
Installed optional packages	106
Using installed optional packages	107
Library reference collection	109
Managing application servers	109
Server collection	110
Application server settings	112
Environment entries collection	121
Starting an application server	122
Directory conventions	123
Restarting an application server in recovery mode	124
Running application servers under specific user profiles	125
Detecting and handling problems with runtime components	126
Stopping an application server	126
Changing time zone settings	127
Changing the ports associated with an application server	146

Web module or application server stops processing requests	147
Preparing to host applications	148
Configuring an application server to use a single network interface	149
Configuring application servers for UCS Transformation Format	151
Directory conventions	152
Managing the QWAS8 subsystem for WebSphere Application Server	153
Starting the application server environment in the QWAS8 subsystem.	153
Configuring application servers to automatically start when the QWAS8 subsystem starts	154
Shutting down the QWAS8 subsystem for WebSphere Application Server	155
Directory conventions	156
Creating generic servers	157
Starting and terminating generic application servers	159
Generic server settings	159
Enabling user profiles to run application servers with System i Navigator	160
Configuring transport chains	160
Transport chains	162
HTTP transport collection	163
HTTP transport settings.	164
Transport chains collection	168
Transport chain settings	169
HTTP tunnel transport channel settings	170
HTTP transport channel settings	170
TCP transport channel settings	175
DCS transport channel settings	178
SSL inbound channel	179
Session Initiation Protocol (SIP) inbound channel settings	180
Session Initiation Protocol (SIP) container inbound channel settings	180
User Datagram Protocol (UDP) Inbound channel settings	181
Web container inbound transport channel settings	182
HTTP transport channel custom properties.	183
HTTP Tunnel transport channel custom properties	185
TCP transport channel custom properties	186
Web container transport chain custom properties	187
Configuring inbound HTTP request chunking	188
Transport chain problems	189
Deleting a transport chain	189
Disabling ports and their associated transport chains	190
Creating custom services	191
Custom service collection	193
Defining application server processes	194
Process definition settings.	195
Automatically restarting server processes	198
Directory conventions	199
Configuring the JVM	200
Java virtual machine settings.	201
Configuring JVM sendRedirect calls to use context root	206
Java virtual machine custom properties	207
Tuning application servers.	245
Tuning the application server using pre-defined tuning templates	247
Web services client to web container optimized communication	251
Appendix. Directory conventions	253
Notices	255
Trademarks and service marks	257

Index 259

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 919-254-5250.

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.

Changes to serve you more quickly

Print sections directly from the information center navigation

PDF books are provided as a convenience format for easy printing, reading, and offline use. The information center is the official delivery format for IBM WebSphere Application Server documentation. If you use the PDF books primarily for convenient printing, it is now easier to print various parts of the information center as needed, quickly and directly from the information center navigation tree.

To print a section of the information center navigation:

1. Hover your cursor over an entry in the information center navigation until the **Open Quick Menu** icon is displayed beside the entry.
2. Right-click the icon to display a menu for printing or searching your selected section of the navigation tree.
3. If you select **Print this topic and subtopics** from the menu, the selected section is launched in a separate browser window as one HTML file. The HTML file includes each of the topics in the section, with a table of contents at the top.
4. Print the HTML file.

For performance reasons, the number of topics you can print at one time is limited. You are notified if your selection contains too many topics. If the current limit is too restrictive, use the feedback link to suggest a preferable limit. The feedback link is available at the end of most information center pages.

Under construction!

The Information Development Team for IBM WebSphere Application Server is changing its PDF book delivery strategy to respond better to user needs. The intention is to deliver the content to you in PDF format more frequently. During a temporary transition phase, you might experience broken links. During the transition phase, expect the following link behavior:

- Links to Web addresses beginning with `http://` work
- Links that refer to specific page numbers within the same PDF book work
- The remaining links will *not* work. You receive an error message when you click them

Thanks for your patience, in the short term, to facilitate the transition to more frequent PDF book updates.

Chapter 1. Configuring port settings

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.

Before you begin

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. Read the "port validator tool" article in the information center for more information.

Tip: Port conflicts might occur if you install WebSphere Application Server on multiple systems with deployment managers managing servers or clusters on different systems. The configuration-service port-resolution mechanism does not support cross profiles on different host machines.

- **Example 1:**

1. On system A, create a cell profile that includes Dmgr and AppSrv01 (Node1).
2. On system B, create AppSrv01 and federate AppSrv01 (Node2) to Dmgr on system A.
3. Create server1 on Node1 and server2 on Node2.
4. The server1 server and server2 server might contain duplicate server endpoint ports in the serverindex.xml file because Node1 and Node2 are located on different host systems.

- **Example 2:**

1. On system A, create a cell profile that includes Dmgr and AppSrv01 (Node1).
2. On system B, create AppSrv01 and federate AppSrv01 (Node2) to Dmgr on system A.
3. On system B, create JobManager.
4. Create a cluster and add two servers, server1 on Node1 and server2 on Node2.
5. The server2 server and the JobManager server might contain duplicate server endpoint ports in the serverindex.xml file because server2 and JobManager are in cross profiles. The server2 server is under Dmgr, JobManager is under the JobManager profile. and the Dmgr and JobManager profiles are located on different machines.

About this task

Procedure

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. Read the "dspwasinst command" article in the information center for more information.

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.

Read the "manageprofiles command" article in the information center for more information.

- You can use the chgwassvr command to change the ports for an application server within a profile.
Read the "chgwassvr command" article in the information center for more information.

Updating ports in existing profiles

Use the updatePorts.ant script to change ports in an installed profile.

Before you begin

The updatePorts.ant script for application server profiles is in the *app_server_root/profileTemplates/template_name/actions* directory. To use the script, you have to identify which profile to update.

Note: You should only run this script if the profile is unfederated and if the configuration is the same structure as it was when the profile was created. For example, this script is ideal for changing ports for an unfederated application server profile after you created the profile but before you altered its configuration. For all other situations, use the techniques described in "Setting port numbers kept in the serverindex.xml file using scripting".

About this task

Use the following procedure to become familiar with using the updatePorts.ant script. Each step is an exercise that results in reassigning ports using a particular method that the updatePorts.ant script supports.

Look at steps for all of the operating systems mentioned. The differences are mainly in the extension of the script file and the direction of the directory delimiters. For example, Linux shell scripts (*.sh) and other commands require a ./ before the command to tell the operating system that the command is in the current working directory.

Chapter 2. Managing profiles on non-z/OS operating systems

You can create and delete profiles, which are sets of files that define the runtime environment. At least one profile must exist to run the product.

Before you begin

This task assumes a basic familiarity with the **manageprofiles** command to create profiles.

About this task

Typically, you create a profile after you install the product. Depending on which WebSphere Application Server product you have, you might create additional profiles.

The Profile Management tool is unavailable. You can use the **manageprofiles** QShell script to create additional profiles. You can also use the IBM® Web Administration for IBM i graphical user interface to create standalone application server profiles.

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 manage profiles.

Procedure

- Create profiles using the **manageprofiles** command.
- Delete profiles.

Results

You might have created or deleted a profile depending on the tasks that you completed.

What to do next

Depending on the action that you completed, you can start a server or proceed to other tasks such as deploying an application.

Profile concepts

A profile defines the runtime environment. The profile includes all the files that the server processes in the runtime environment and that you can change.

You can create a runtime environment through the **manageprofiles** command. Depending on the operation that you want to perform with the **manageprofiles** command, you need to provide one or more parameters. You can use the command to do such actions as creating or deleting profiles. To create a cell profile, you must invoke the **manageprofiles** command two separate times.

Core product files

The core product files are the shared product binary files, which are shared by all profiles.

The directory structure for the product has the following 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 *app_server_root/profiles* directory is the default directory for creating profiles.

When you want binary files at different service levels, you must use a separate installation of the product 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 create a profile in an installation root directory, then 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 profile for the product.

Run the command-line tool each time that you want to create a profile.

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 a single set of product core files. Also, creating new profiles is more efficient 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 profile on the same machine as an existing profile. 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.

Multiple directories exist within this directory, which correspond to different profile types and vary with the type of product that is 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 exist outside the *profileTemplates* directory, if you have any.

See the `-templatePath` parameter description in the **manageprofiles** command topic for more information.

The **manageprofiles** command can create the following type of profile:

Management profile with an administrative agent server

The basic function of the administrative agent is to provide a single interface to administer multiple application servers.

Specify management for the `-templatePath` parameter and `ADMIN_AGENT` for the `-serverType` parameter to create this type of management profile with the **manageprofiles** command.

Application server profile

Use the application server to make applications available to the Internet or to an intranet.

An important product feature is the ability to scale up a standalone 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 standalone application server.

Each standalone application server can optionally have 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 standalone application server node unless you decide to add the application server node to a deployment manager cell. Adding the application server node to a cell is known as *federation*. Federation changes the standalone application server node 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, then use the administrative console and the scripting interface of the standalone application server node to manage the process.

The application server profile is created by default if you do not specify the `-templatePath` parameter. You can alternatively specify `default` for the `-templatePath` parameter on the **manageprofiles** command to create the application server 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. You can use the `-profileName` parameter with most of the scripts to enable the scripts to act on a profile other than the default profile.

After installation, you should use the **manageprofiles** command to create a default profile named `default`. The default profile should be a standalone application server profile containing a single application server named `server1`.

Addressing a profile in a multiprofile environment: When multiple profiles exist on a machine, certain commands require that you specify the `-profileName` parameter if the profile is not the default profile. In those cases, it might be easier to use the commands that are in the `bin` directory of each profile. When you issue one of these commands within the `bin` directory of a profile, the command acts on that profile unless the `-profileName` parameter specifies a different profile.

Security policy for application server profiles

In environments where you plan to have multiple standalone 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 are 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 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. Different profile types might include different subdirectories. This example assumes that the profile, 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/temp`
- `user_data_root/profiles/AppSrv01/wstemp`

Profiles: File-system requirements

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.

Table 1. Space requirements.

This table shows space requirements for various profiles and server types.

Profile or server type	Space required
Application server	200 MB
Management	30 MB

Situations in which you could have insufficient file-system space

The **manageprofiles** command checks that the amount of file-system space needed to create the profile is available right before profile creation begins. However, a slight chance exists that the profile creation can fail due to a lack of file-system space. This failure can occasionally occur in the following situations:

- Another user performs an action, such as copying files, that occupies file-system space at the same time that the **manageprofiles** command writes to the file system.
- Another program writes to the disk at the same time that the **manageprofiles** command writes to it to create a profile.
- The **manageprofiles** command writes its logs and the profile that it creates to the same file system at the same time.

Use the following recommendations to avoid profile creation failure:

- Ensure that enough temporary space is allocated for profile creation. Some temporary space is needed for the profile creation logs. These logs can be on a different file system than the file system on which the profile is created.
- Ensure no other program writes to the file-system space when the **manageprofiles** command creates the profile.
- Ensure no user performs actions that occupy the file-system space when the **manageprofiles** command creates the profile.

Managing profiles using commands

Use commands to create a profile, start the server of the profile, display ports used by your server, and open the administrative console.

Before you begin

This task assumes a basic familiarity with the command, other application server commands, and system commands.

Before you can create and use a profile, you must install the product.

About this task

Perform the following steps to create a profile, start the server of the profile, display ports used by your server, and open the administrative console for your server.

This example deals with the profile environment of a standalone application server.

Procedure

1. Create the server profile from the original installation:

- `app_server_root/bin/manageprofiles`

Assume that you create the profile by using the defaults. The following script is an example for creating an application server profile:

- `app_server_root/bin/manageprofiles.sh -create -templatePath app_server_root/profileTemplates/default`

2. Change directories to the `profile_root/bin` directory of the new server profile.
3. Start the server.

Issue the `startServer` command.

The server name is the same name as the profile, which, in this case, is `profile_name`.

```
startServer -profileName profile_name
```

Note: The `-profileName` argument is not necessary if you have already changed to the `profile_root/bin` directory of the target profile.

4. Display the ports.

These are the ports assigned during profile creation.

Use the `dspwasinst QShell` script to display the ports for your new profile:

```
app_server_root/bin/dspwasinst -profileName profileName
```

The `WC_adminhost` and `WC_adminhost_secure` ports listed are the nonsecure and secure administrative console ports, respectively.

5. Open the administrative console.

The `server1` administrative console is defined on the `WC_adminhost` setting for the non-secure administrative console port or the `WC_adminhost_secure` setting for the secure administrative console port.

If the value of the `WC_adminhost` port for your server is 20003, for example, specify the following web address in your browser:

```
http://host_name_or_IP_address:20003/ibm/console/
```

If the value of the `WC_adminhost_secure` port for your server is 9061, for example, specify the following web address in your browser:

```
https://host_name_or_IP_address:9061/ibm/console/
```

Results

You created an application server profile, started an application server, and accessed the administrative console using your browser.

What to do next

Deploy an application.

manageprofiles command

Use the `manageprofiles` command to create, delete, augment, back up, and restore profiles, which define runtime environments. Using profiles instead of multiple product installations saves disk space and simplifies updating the product because a single set of core product files is maintained.

The command file is located in the `app_server_root/bin` directory. The command file is a script named `manageprofiles`.

Remember: If you use this command with the managed profile template, application servers are not created. However, ports are still used if you are federating a node.

Syntax

The `manageprofiles` command is used to perform the following tasks:

- create a profile (-create)
- delete a profile (-delete)
- augment a profile (-augment)
- unaugment a profile (-unaugment)
- unaugment all profiles that have been augmented with a specific augmentation template (-unaugmentAll)
- delete all profiles (-deleteAll)
- list all profiles (-listProfiles)
- list augments for a profile (-listAugments)
- get a profile name (-getName)
- get a profile path (-getPath)
- validate a profile registry (-validateRegistry)
- validate and update a profile registry (-validateAndUpdateRegistry)
- get the default profile name (-getDefaultName)
- set the default profile name (-setDefaultName)
- back up a profile (-backupProfile)
- restore a profile (-restoreProfile)
- perform `manageprofiles` command tasks that are contained in a response file (-response)

For detailed help including the required parameters for each of the tasks accomplished with the `manageprofiles` command, use the `-help` parameter. The following example uses the help parameter with the `manageprofiles -augment` command on Windows operating systems:

```
app_server_root\bin\manageprofiles.bat -augment -help
```

The output from the help command will specify which parameters are required and which are optional.

Depending on the operation that you want to perform with the `manageprofiles` command, you need to provide one or more of the following parameters. The command-line tool validates that the required

parameters are provided and the values entered for those parameters are valid. Be sure to type the name of the parameters with the correct upper and lower case as the command-line tool does not validate the case of the parameter name. Incorrect results can occur when the parameter case is not typed correctly.

- `-profileName` *profile_name*
- `-profilePath` *profile_root*
- `-templatePath` *template_path*
- `-nodeName` *node_name*
- `-cellName` *cell_name*
- `-hostName` *host_name*
- `-serverName` *server_name*
- `-adminUserName` *adminUser_ID*
- `-adminPassword` *adminPassword*
- `-backupFile` *backupFile_name*
- `-debug`
- `-enableAdminSecurity` *true | false*
- `-federateLater` *true | false*
- `-importPersonalCertKS` *keystore_path*
- `-importPersonalCertKSType` *keystore_type*
- `-importPersonalCertKSPassword` *keystore_password*
- `-importPersonalCertKSAlias` *keystore_alias*
- `-importSigningCertKS` *keystore_path*
- `-importSigningCertKSType` *keystore_type*
- `-importSigningCertKSPassword` *keystore_password*
- `-importSigningCertKSAlias` *keystore_alias*
- `-isDefault`
- `-isDeveloperServer`
- `-applyPerfTuningSetting` *standard | production | development*
- `-keyStorePassword` *keystore_password*
- `-listAugments`
- `-omitAction` *feature1 feature2... featureN*
- `-personalCertDN` *distinguished_name*
- `-personalCertValidityPeriod` *validity_period*
- `-response` *response_file*
- `-serverType` *ADMIN_AGENT*
- `-signingCertDN` *distinguished_name*
- `-signingCertValidityPeriod` *validity_period*
- `-startingPort` *starting_port* | `-portsFile` *file_path* | `-defaultPorts`
- `-unaugmentAll`
- `-unaugmentDependents` *true | false*
- `-validatePorts`
- `-webServerCheck` *true | false*
- `-webServerHostname` *webserver_host_name*
- `-webServerInstallPath` *webserver_installpath_name*
- `-webServerName` *webserver_name*
- `-webServerOS` *webserver_operating_system*

- `-webServerPluginPath` *webserver_plugin_path*
- `-webServerPort` *webserver_port*
- `-webServerType` *webserver_type*

The following example uses the `manageprofiles -create` command on operating systems such as AIX® or Linux:

```
app_server_root/bin/manageprofiles.sh -create
  -profileName profile_name
  -profilePath profile_root
  -templatePath template_path
```

Parameters

The following options are available for the `manageprofiles` command:

-adminUserName *adminUser_ID*

Specify the user ID that is used for administrative security.

-adminPassword *adminPassword*

Specify the password for the administrative security user ID specified with the `-adminUserName` parameter.

-augment

Use the `augment` parameter to make changes to an existing profile with an augmentation template. The `augment` parameter causes the `manageprofiles` command to update or augment the profile identified in the `-profileName` parameter using the template in the `-templatePath` parameter. The augmentation templates that you can use are determined by which IBM products and versions are installed in your environment.

Important: The templates that are included with the WebSphere Application Server base product can only be used to create profiles and not to augment existing profiles because only create templates are shipped with the product.

Also, do not manually modify the files that are located in the `install_dir/profileTemplates` directory. For example, if you are changing the ports during profile creation, use the `-startingPort` or `-portsFile` arguments on the `manageprofiles` command instead of modifying the file in the profile template directory.

You can specify a relative path for the `-templatePath` parameter if the profile templates are relative to the `app_server_root/profileTemplates` directory. Otherwise, specify the fully qualified template path. For example:

```
manageprofiles -augment -profileName profile_name -templatePath template_path
```

See also the `-unaugment` parameter.

-backupProfile

Performs a file system backup of a profile folder and the profile metadata from the profile registry file. Any servers using the profile that you want to back up must first be stopped prior to invoking the `manageprofiles` command with the `-backupProfile` option. The `-backupProfile` parameter must be used with the `-backupFile` and `-profileName` parameters, for example:

```
manageprofiles(.bat)(.sh) -backupProfile -profileName profile_name -backupFile backupFile_name
```

When you back up a profile using the `-backupProfile` option, you must first stop the server and the running processes for the profile that you want to back up.

-backupFile *backupFile_name*

Backs up the profile registry file to the specified file. You must provide a fully qualified file path for the `backupFile_name`.

-cellName *cell_name*

Specifies the cell name of the profile. Use a unique cell name for each profile.

This is an optional parameter. If you omit the parameter, a default cell name is assigned.

The default cell names are as follows:

- dmgr template: *profilenameNetwork*
- default template: *shorthostname_profilename*
- managed template: *shorthostname_profilename*
- cell template: Same as the previous dmgr example for the two profiles that are created.

The value for this parameter must not contain spaces or any invalid characters that are not valid such as the following: *, ?, ", <, >, ,, /, \, |, and so on.

-create

Creates the profile.

Specify manageprofiles -create -templatePath *fully_qualified_file_path_to_template* -help for specific information about creating a profile. Available templates include:

- management - Management. Use in conjunction with the -serverType parameter to indicate the type of management profile.
- default - Application server

-debug

Turns on the debug function of the Ant utility, which the manageprofiles command uses.

-personalCertValidityPeriod *validity_period*

An optional parameter that specifies the amount of time in years that the default personal certificate is valid. If you do not specify this parameter with the -personalCertDN parameter, the default personal certificate is valid for one year.

-defaultPorts

Assigns the default or base port values to the profile.

Do not use this parameter when using the -startingPort or -portsFile parameter.

During profile creation, the manageprofiles command uses an automatically generated set of recommended ports if you do not specify the -startingPort parameter, the -defaultPorts parameter or the -portsFile parameter. The recommended port values can be different than the default port values based on the availability of the default ports.

Remember: Do not use this parameter if you are using the managed profile template.

-delete

Deletes the profile.

The profile directory is deleted when you delete the profile so that you can recreate the profile without having to manually delete the directory.

If you delete a profile that has augmenting templates registered to it in the profile registry, then unaugment actions are performed automatically.

gotcha: If you are deleting an old node that has been migrated, shut down the new migrated deployment manager before deleting the old node. This will ensure that the new migrated node is not accidentally removed from the new migrated cell.

-deleteAll

Deletes all registered profiles.

The directory for the profile is deleted when you delete the profile so that when you recreate the profile, you do not have outdated information to manage.

If you delete a profile that has augmenting templates registered to it in the profile registry, then unaugment actions are performed automatically.

-enableAdminSecurity true | false

Enables administrative security. Valid values include true or false. The default value is false.

When enableAdminSecurity is set to true, you must also specify the parameters -adminUserName and -adminPassword along with the values for these parameters.

You cannot use the -enableAdminSecurity parameter to enable administrative security for a custom profile. For security to be enabled for a custom profile, the custom profile must be federated into a deployment manager. Administrative security enabled for the deployment manager is required to enable security for the federated custom profile.

-federateLater true | false

Indicates if the managed profile will be federated during profile creation or if you will federate it later using the addNode command. If the dmgrHost, dmgrPort, dmgrAdminUserName and dmgrAdminPassword parameters do not have values, the default value for this parameter is true. Valid values include true or false.

-getDefaultName

Returns the name of the default profile.

-getName

Gets the name for a profile registered at a given -profilePath parameter.

-getPath

Gets the file system location for a profile of a given name. Requires the -profileName parameter.

-help

Displays command syntax.

-hostName *host_name*

Specifies the host name where you are creating the profile. This should match the host name that you specified during installation of the initial product. The default value for this parameter is the long form of the domain name system. The value for this parameter must be a valid IPv6 host name and must not contain spaces or any characters that are not valid such as the following: *, ?, ", <, >, ,, /, \, |, and so on.

-ignoreStack

An optional parameter that is used with the -templatePath parameter to unaugment a particular profile that has been augmented. See the -unaugment parameter.

-importPersonalCertKS *keystore_path*

Specifies the path to the keystore file that you use to import a personal certificate when you create the profile. The personal certificate is the default personal certificate of the server.

Note: When you import a personal certificate as the default personal certificate, import the root certificate that signed the personal certificate. Otherwise, the manageprofiles command adds the public key of the personal certificate to the trust.p12 file and creates a root signing certificate.

The -importPersonalCertKS parameter is mutually exclusive with the -personalCertDN parameter. If you do not specifically create or import a personal certificate, one is created by default.

When you specify any of the parameters that begin with -importPersonal, you must specify them all.

-importPersonalCertKSType *keystore_type*

Specifies the type of the keystore file that you specify on the -importPersonalCertKS parameter. Values might be JCEKS, CMSKS, PKCS12, PKCS11, and JKS. However, this list can change based on the provider in the java.security file.

When you specify any of the parameters that begin with -importPersonal, you must specify them all.

-importPersonalCertKSPassword *keystore_password*

Specifies the password of the keystore file that you specify on the -importPersonalCertKS parameter.

When you specify any of the parameters that begin with `-importPersonal`, you must specify them all.

-importPersonalCertKSAlias *keystore_alias*

Specifies the alias of the certificate that is in the keystore file that you specify on the `-importPersonalCertKS` parameter. The certificate is added to the server default keystore file and is used as the server default personal certificate.

When you specify any of the parameters that begin with `-importPersonal`, you must specify them all.

-importSigningCertKS *keystore_path*

Specifies the path to the keystore file that you use to import a root certificate when you create the profile. The root certificate is the certificate that you use as the server default root certificate. The `-importSigningCertKS` parameter is mutually exclusive with the `-signingCertDN` parameter. If you do not specifically create or import a root signing certificate, one is created by default.

When you specify any of the parameters that begin with `-importSigning`, you must specify them all.

-importSigningCertKSType *keystore_path*

Specifies the type of the keystore file that you specify on the `-importSigningCertKS` parameter. Valid values might be JCEKS, CMSKS, PKCS12, PKCS11, and JKS. However, this list can change based on the provider in the `java.security` file.

When you specify any of the parameters that begin with `-importSigning`, you must specify them all.

-importSigningCertKSPassword *keystore_password*

Specifies the password of the keystore file that you specify on the `-importSigningCertKS` parameter.

When you specify any of the parameters that begin with `-importSigning`, you must specify them all.

-importSigningCertKSAlias *keystore_alias*

Specifies the alias of the certificate that is in the keystore file that you specify on the `-importSigningCertKS` parameter. The certificate is added to the server default root keystore and is used as the server default root certificate.

When you specify any of the parameters that begin with `-importSigning`, you must specify them all.

-isDefault

Specifies that the profile identified by the accompanying `-profileName` parameter is to be the default profile once it is registered. When issuing commands that address the default profile, it is not necessary to use the `-profileName` attribute of the command.

-isDeveloperServer

Specifies that the server is intended for development purposes only. This parameter is useful when creating profiles to test applications on a non-production server before deploying the applications on their production application servers.

This parameter is valid only for the default profile template.

If you specify both the `-isDeveloperServer` and `-applyPerfTuningSetting` parameters, depending on the option selected for `-applyPerfTuningSetting`, `-applyPerfTuningSetting` might override `-isDeveloperServer`.

-applyPerfTuningSetting *option*

Specifies the performance-tuning setting that most closely matches the type of environment in which the application server will run.

This parameter is only valid for the default profile template.

standard

The standard settings are the standard out-of-the-box default configuration settings that are optimized for general-purpose usage.

production

The production performance settings are optimized for a production environment where application changes are rare and optimal runtime performance is important.

development

The development settings are optimized for a development environment where frequent application updates are performed and system resources are at a minimum.

Important: Do not use the development settings for production servers.

If you specify both the **-isDeveloperServer** and **-applyPerfTuningSetting** parameters, depending on the option selected for **-applyPerfTuningSetting**, **-applyPerfTuningSetting** might override **-isDeveloperServer**.

-keyStorePassword *keystore_password*

Specifies the password to use on all keystore files created during profile creation. Keystore files are created for the default personal certificate and the root signing certificate.

-listAugments

Lists the registered augments on a profile that is in the profile registry. You must specify the **-profileName** parameter with the **-listAugments** parameter.

-nodeName *node_name*

Specifies the node name for the node that is created with the new profile. Use a unique value on the machine. Each profile that shares the same set of product binaries must have a unique node name.

The following default node names exist:

- management template for the deployment manager, administrative agent, or the job manager:
profilenameManager
- default template: *shorthostname_profilename*
- managed template: *shorthostname_profilename*
- cell: See the previous management and default template examples and apply as appropriate to the two profiles that are created.
- secureproxy template: *shorthostname_profilename*

The value for this parameter must not contain spaces or any characters that are not valid such as the following: *, ?, ", <, >, ,, /, \, |, and so on.

-omitAction *feature1 feature2... featureN*

An optional parameter that excludes profile features.

Each profile template comes predefined with certain optional features. The following optional features can be used with the **-omitAction** parameter for the following profile templates:

- default - Application server
 - deployAdminConsole
 - defaultAppDeployAndConfig

-personalCertDN *distinguished_name*

Specifies the distinguished name of the personal certificate that you are creating when you create the profile. Specify the distinguished name in quotes. This default personal certificate is located in the server keystore file. The **-importPersonalCertKSType** parameter is mutually exclusive with the **-personalCertDN** parameter. See the **-personalCertValidityPeriod** parameter and the **-keyStorePassword** parameter.

-portsFile *file_path*

An optional parameter that specifies the path to a file that defines port settings for the new profile.

Do not use this parameter when using the **-startingPort** or **-defaultPorts** parameter.

During profile creation, the **manageprofiles** command uses an automatically generated set of recommended ports if you do not specify the **-startingPort** parameter, the **-defaultPorts** parameter or the **-portsFile** parameter. The recommended port values can be different than the default port values based on the availability of the default ports.

-profileName *profile_name*

Specifies the name of the profile. Use a unique value when creating a profile. Each profile that shares the same set of product binaries must have a unique name. The default profile name is based on the profile type and a trailing number, for example:

`<profile_type><profile_number>`

where

- `<profile_type>` is a value such as AppSrv, Dmgr, AdminAgent, JobMgr, or Custom
- `<profile_number>` is a sequential number that creates a unique profile name

The value for this parameter must not contain spaces or characters that are not valid such as any of the following: *, ?, ", <, >,,, /, \, |, and so on.

The profile name that you choose must not be in use.

-profilePath *profile_root*

Specifies the fully qualified path to the profile, which is referred to as the *profile_root*.

The default value is based on the *user_data_root* directory, the profiles subdirectory, and the name of the profile.

For example, the default is:

`WS_WSPROFILE_DEFAULT_PROFILE_HOME/profileName`

The `WS_WSPROFILE_DEFAULT_PROFILE_HOME` element is defined in the `wasprofile.properties` file in the `app_server_root/properties` directory.

The value for this parameter must be a valid path for the target system and must not be currently in use.

The QEJBSVR profile must have permissions to write to the directory.

-response *reponse_file*

Accesses all API functions from the command line using the `manageprofiles` command.

The command line interface can be driven by a response file that contains the input arguments for a given command in the properties file in key and value format. Use the following example response file to run a create operation:

```
create
profileName=testResponseFileCreate
profilePath=profile_root
templatePath=app_server_root/profileTemplates/default
nodeName=myNodeName
cellName=myCellName
hostName=myHostName
omitAction=myOptionalAction1,myOptionalAction2
```

To determine which input arguments are required for the various types of profile templates and action, use the `manageprofiles` command with the `-help` parameter.

-restoreProfile

Restores a profile backup. Must be used with the `-backupFile` parameter, for example:

`manageprofiles(.bat)(.sh) -restoreProfile -backupFile file_name`

To restore a profile, perform the following steps:

1. Stop the server and the running processes for the profile that you want to restore.
2. Manually delete the directory for the profile from the file system.
3. Run the `-validateAndUpdateRegistry` option of the `manageprofiles` command.
4. Restore the profile by using the `-restoreProfile` option of the `manageprofiles` command.

-serverName *server_name*

Specifies the name of the server. Specify this parameter only for the default and secureproxy

templates. If you do not specify this parameter when using the default or secureproxy templates, the default server name is server1 for the default profile, and proxy1 for the secure proxy profile.

-serverType ADMIN_AGENT

Specifies the type of management profile. Specify ADMIN_AGENT for an administrative agent server. This parameter is required when you create a management profile.

-setDefaultName

Sets the default profile to one of the existing profiles. Must be used with the -profileName parameter, for example:

```
manageprofiles(.bat)(.sh) -setDefaultName -profileName profile_name
```

-signingCertDN distinguished_name

Specifies the distinguished name of the root signing certificate that you create when you create the profile. Specify the distinguished name in quotes. This default personal certificate is located in the server keystore file. The -importSigningCertKS parameter is mutually exclusive with the -signingCertDN parameter. If you do not specifically create or import a root signing certificate, one is created by default. See the -signingCertValidityPeriod parameter and the -keyStorePassword.

-signingCertValidityPeriod validity_period

An optional parameter that specifies the amount of time in years that the root signing certificate is valid. If you do not specify this parameter with the -signingCertDN parameter, the root signing certificate is valid for 15 years.

-startingPort startingPort

Specifies the starting port number for generating and assigning all ports for the profile.

Port values are assigned sequentially from the -startingPort value.

Do not use this parameter with the -defaultPorts or -portsFile parameters.

During profile creation, the manageprofiles command uses an automatically generated set of recommended ports if you do not specify the -startingPort parameter, the -defaultPorts parameter or the -portsFile parameter. The recommended port values can be different than the default port values based on the availability of the default ports.

Attention: Do not use this parameter if you are using the managed profile template.

-templatePath template_path

Specifies the directory path to the template files in the installation root directory. Within the profileTemplates directory are various directories that correspond to different profile types and that vary with the type of product installed. The profile directories are the paths that you indicate while using the -templatePath option. You can specify profile templates that lie outside the installation root, if you happen to have any.

The default template path is *app_server_root/profileTemplates/default*. You can use a relative path for the -templatePath parameter. The path is relative to the current working directory or to *app_server_root/profileTemplates*, in that order. The following example creates a profile based on the default for a standalone application server:

```
manageprofiles -create -profileName MyProfile -startingPort 10380
```

-unaugment

Augmentation is the ability to change an existing profile with an augmentation template. To unaugment a profile that has been augmented, you must specify the -unaugment parameter and the -profileName parameter. If a series of manageprofiles augmentations were performed, and you specify only these two parameters to unaugment a profile, the unaugment action undoes the last augment action first.

To unaugment a particular profile that has been augmented, additionally specify the -ignoreStack parameter with the -templatePath parameter. Normally, you would not unaugment a particular profile because you must ensure that you are not violating profile template dependencies.

When using the -templatePath parameter, you can specify a relative file path for the parameter.

See also the `augment` parameter.

-augmentAll

Unaugments all profiles that have been augmented with a specific augmentation template. The `-templatePath` parameter is required with the `-augmentAll` parameter.

When using the `-templatePath` parameter, you can specify a relative file path for the parameter.

Optionally, specify the `-augmentDependents` parameter with the `-augmentAll` parameter to unaugment all profiles that are prerequisites of the profiles that are being unaugmented.

Note: If you use this parameter when you have no profiles augmented with the profile templates, an error might be delivered.

See also the `augment` parameter.

-augmentDependents true | false

If set to `true`, the parameter unaugments all the augmented profiles that are prerequisites to the profiles being unaugmented with the `-augmentAll` parameter. The default value for this parameter is `false`.

Optionally specify the `-augmentDependents` parameter with the `-augmentAll` parameter.

-validateAndUpdateRegistry

Checks all of the profiles that are listed in the profile registry to see if the profiles are present on the file system. Removes any missing profiles from the registry. Returns a list of the missing profiles that were deleted from the registry.

-validateRegistry

Checks all of the profiles that are listed in the profile registry to see if the profiles are present on the file system. Returns a list of missing profiles.

-validatePorts

Specifies the ports that should be validated to ensure they are not reserved or in use. This parameter helps you to identify ports that are not being used. If a port is determined to be in use, the profile creation stops and an error message displays. You can use this parameter at any time on the create command line. It is recommended to use this parameter with the `-portsFile` parameter.

-webServerCheck true | false

Indicates if you want to set up web server definitions. Valid values include `true` or `false`. The default value for this parameter is `false`.

-webServerHostname *webserver_host_name*

The host name of the server. The default value for this parameter is the long host name of the local machine.

-webServerInstallPath *webserver_installpath_name*

The installation path of the web server, local or remote. The default value for this parameter is dependent on the operating system of the local machine and the value of the `webServerType` parameter. For example:

-webServerName *webserver_name*

The name of the web server. The default value for this parameter is `webserver1`.

-webServerOS *webserver_operating_system*

The operating system from where the web server resides. Valid values include: `windows`, `linux`, `solaris`, `aix`, `hpux`, `os390`, and `os400`. Use this parameter with the `webServerType` parameter.

-webServerPluginPath *webserver_pluginpath*

The path to the plug-ins that the web server uses. The default value for this parameter is `WAS_HOME/plugins`.

-webServerPort *webserver_port*

Indicates the port from where the web server will be accessed. The default value for this parameter is 80.

-webServerType *webserver_type*

The type of the web server. Valid values include: IHS, SUNJAVASYSTEM, IIS, DOMINO, APACHE, and HTTPSERVER_ZOS. Use this parameter with the webServerOS parameter.

Usage scenario

The following examples demonstrate correct syntax. Issue the command in any of the following examples on one line. Each example shows the command on more than one line to increase clarity.

- Creating an application server profile

Create an application server profile named Default01 with the following command.

```
manageprofiles -create
-profileName Default01
-templatePath default
-startingPort 21000
-personalCertDN "cn=testa, ou=Rochester, o=IBM, c=US"
-signingCertDN "cn=testc, ou=Rochester, o=IBM, c=US"
-keyStorePassword ap3n9krw
```

Logs

The manageprofiles command creates a log for every profile that it creates.

- The logs are in the *user_data_root/profileRegistry/logs/manageprofiles* directory. The files are named in this pattern: *profile_name_create.log*.
- The command also creates a log for every profile that it deletes. The logs are in the *user_data_root/profileRegistry/logs/manageprofiles* directory. The files are named in this pattern: *profile_name_delete.log*.

Example: Incrementing default port numbers from a starting point

The manageprofiles command can assign port numbers based on a starting port value. You can provide the starting port value from the command line, using the `-startingPort` parameter. The command assigns port numbers sequentially from the starting port number value. However, if a port value in the sequence conflicts with an existing port assignment, the next available port value is used.

The order of port assignments is arbitrary. Predicting assignments is not possible.

For example, ports created with `-startingPort 20002` would appear similar to the following example:

Assigned ports for an application server profile

```
WC_defaulthost=20002
WC_adminhost=20003
WC_defaulthost_secure=20004
WC_adminhost_secure=20005
BOOTSTRAP_ADDRESS=20006
SOAP_CONNECTOR_ADDRESS=20007
IPC_CONNECTOR_ADDRESS=20008
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=20009
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=20010
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=20011
ORB_LISTENER_ADDRESS=20012
CELL_DISCOVERY_ADDRESS=20013
NODE_MULTICAST_DISCOVERY_ADDRESS=20014
NODE_IPV6_MULTICAST_DISCOVERY_ADDRESS=20015
NODE_DISCOVERY_ADDRESS=20016
DCS_UNICAST_ADDRESS=20017
SIB_ENDPOINT_ADDRESS=20018
SIB_ENDPOINT_SECURE_ADDRESS=20019
SIB_MQ_ENDPOINT_ADDRESS=20020
SIB_MQ_ENDPOINT_SECURE_ADDRESS=20021
SIP_DEFAULTHOST=20022
SIP_DEFAULTHOST_SECURE=20023
```

The following example uses the startingPort parameter of the manageprofiles command and creates ports from an initial value of 20002, with the content shown in the previous example:

```
app_server_root/bin/manageprofiles -create
    -profileName shasti
    -profilePath user_data_root/profiles/shasti
    -templatePath app_server_root/profileTemplates/default
    -nodeName W2K03
    -cellName W2K03_Cell01
    -hostName planeEnt
    -startingPort 20002
```

Example: Using predefined port numbers

The manageprofiles command recommends initial port values when you do not explicitly set port values. You can use predefined port values instead.

The manageprofiles command recommends port values when the options of -defaultPorts, -startingPort, or -portsFile are not specified.

Table 2. File locations of default port values.

This table lists the file locations of default port values by type of profile.

Profile	File path
Application server	app_server_root/profileTemplates/default/actions/portsUpdate/portdef.props
Management profile for an administrative agent server	app_server_root/profileTemplates/management/actions/portsUpdate/adminagent.portdef.props

To customize the port values in the portdef.props file before creating your profile, perform the following steps. The following example creates the default profile. For other types of profiles, you must substitute the file path with the file path of the profile that you want to create.

1. Copy the `app_server_root/profileTemplates/default/actions/portsUpdate/portdef.props` file from the default profile template path and place a copy of the file in an arbitrary temporary directory such as:
 - /temp/ports
2. In the new file, modify the port settings to specify your port values.
3. Create your profile with the manageprofiles command. Use the modified port values. Specify the location of your modified portdef.props file on the -portsFile parameter. Specify the -validatePorts parameter to ensure that ports are not reserved or in use. Use the following example as a guide:

```
manageprofiles
    -create
    -profileName Wow_Profile
    -profilePath profile_root
    -templatePath app_server_root\profileTemplates\default
    -nodeName Wow_node
    -cellName Wow_cell
    -hostName lorriemb
    -portsFile \temp\ports\portdef.props
    -validatePorts
```

Suppose that the portdef.props file has the following values:

```
WC_defaulthost=39080
WC_adminhost=39060
WC_defaulthost_secure=39443
WC_adminhost_secure=39043
BOOTSTRAP_ADDRESS=32809
SOAP_CONNECTOR_ADDRESS=38880
IPC_CONNECTOR_ADDRESS=39633
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=39401
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=39403
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=39402
ORB_LISTENER_ADDRESS=39100
DCS_UNICAST_ADDRESS=39353
SIB_ENDPOINT_ADDRESS=37276
SIB_ENDPOINT_SECURE_ADDRESS=37286
```

```
SIB_MQ_ENDPOINT_ADDRESS=35558
SIB_MQ_ENDPOINT_SECURE_ADDRESS=35578
SIP_DEFAULTHOST=35060
SIP_DEFAULTHOST_SECURE=35061
```

After running the `manageprofiles` command to create your profile with the user defined port values, a success or fail result displays.

Note: After you create a profile successfully, the console prints a message that indicates success and advises you to check the `AboutThisProfile.txt` file. However, a `AboutThisProfile.txt` file is not generated when you create a client profile or a plug-ins profile on IBM i.

The `manageprofiles` command creates a copy of the current `portdefs.props` file in the `profile_root\properties` directory.

Use only one of the three port values parameters, `-startingPort`, `-defaultPorts`, or `-portsFile` with the `manageprofiles` command. The three parameters are mutually exclusive.

Managing default profiles

You can create default profiles for the operating system. Scripts that do not specify a profile use the default profile.

About this task

Default profiles are included with the product. A profile is a set of configuration files and installed applications that make up your WebSphere Application Server environment.

Procedure

- Install the web server plug-in feature without the core product feature to create a default remote HTTP profile.
- Install the secure proxy server feature without the core product feature to create a default secure proxy 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.

Results

Depending on the steps that you performed, you have created at least one default profile.

What to do next

Continue configuring your environment if you have not completed the configuration.

Default application server profiles

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 server, for example, *MYISERIES*. The cell name is case-sensitive.

Node name

The host name of the server, for example, *MYISERIES*. The node name is case-sensitive.

Application server name

server1. The application server name is case-sensitive.

Predeployed 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* example 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.

Default ports

The default profile is configured to use the ports that are listed in the Port number settings article in the information center. 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 article on the *chgwassvr* command in the information center. 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 article on the *dspwasinst* command in the information center.

Default secure proxy profiles

The name of the default secure proxy profile is *SecureProxySrv01*. This profile is created when you install the secure proxy server feature.

The default secure proxy profile is contained under the following directory structure:

```
user_data_root/profiles/SecureProxySrv01
```


The default secure proxy profile does not contain an application server: this profile contains a secure proxy server named *proxy1*. The *proxy1* server runs on a workstation in the DMZ, whereas the application server runs on a workstation in the secure zone.

Default application client profiles

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 the application client product.

In an application client configuration, the application client and the application server can run on separate machines or logical partitions.

Default remote HTTP profiles

The name of the default remote HTTP profile is *http*. This profile is created when you install the web server plug-in.

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.

Deleting profiles

You can delete a profile using the `manageprofiles` command. If the command fails, you can delete the profile using operating system commands.

Before you begin

Before you delete a profile, stop its application server to ensure that the application server can be deleted.

You cannot delete a profile using the Profile Management Tool.

About this task

The following example attempts to delete a profile using the `manageprofiles` command, and then using operating system commands.

Procedure

1. Issue the `manageprofiles` command to delete a profile.

Substitute your profile name for the *profile_name* value in the following commands.

```
./manageprofiles -delete  
-profileName profile_name
```

If the command is successful, you have completed the task and can skip the remaining steps. If the command is partially successful or unsuccessful, proceed to the next step to delete the profile manually. If you receive the `INSTCONFFAILED: Cannot delete profile.` message, the command was unsuccessful. If the deletion is partially successful, you could receive message information similar to the following wording:

INSTCONFPARTIALSUCCESS: The profiles no longer exist, but errors occurred.
For more information, consult
app_server_root/logs/manageprofiles/deleteAll.log.

2. Issue operating system commands to delete the profile directory.
3. Issue the following command to remove references in the registry to deleted profiles:
`manageprofiles -validateAndUpdateRegistry`

Editing of the registry is not recommended.

Results

You have now deleted a profile.

What to do next

You can delete other profiles using this procedure, or create other profiles using the `manageprofiles` command.

Chapter 3. Administering nodes and resources

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.

About this task

You can administer stand-alone application servers using an administrative agent. If your system uses administrative services, you can specify settings for the service.

Procedure

- Administer stand-alone application servers on the same computer using an administrative agent.
- Use the settings page for an administrative service to configure administrative services.
- Change the host name.

What to do next

Administer nodes and node resources as needed using the administrative console or other administrative tools.

Working with nodes - groups of managed servers

A node is a grouping of managed or unmanaged servers. You can add both managed and unmanaged nodes to your product 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 node in the topology for managing web servers or servers other than WebSphere application servers, you add an unmanaged node. You can add, configure, remove, and otherwise work with nodes, node agents, and node groups.

Changing host names

After creating a profile, the host name of the server or its ports might be incorrect. You can follow the examples to change the server host name using command line tools and the wsadmin scripting tool, and the host names of the server ports using the administrative console and command line tools.

Before you begin

Create a profile. Verify that the host name of the server and the server ports are correct.

About this task

If the host name of a server or its ports is incorrect, then you might experience problems such as errors when you attempt to stop a server. One example task shows how to correct the server host name through command line tools and the wsadmin scripting tool. The other example task shows how to correct the host name of the server ports using the administrative console and command line tools.

Procedure

- Correct the host name for an application server using the wsadmin scripting tool and command line tools.
 1. Launch the wsadmin tool.
Enter the following command:

```
wsadmin -lang jython
```
 2. List the contents of the server configuration file.
Enter the following line of code:

```
AdminConfig.list('ServerIndex')
```

3. In the output, find the `ServerIndex` object for the application server, similar to the following example:

```
cells/isthmusCell116/nodes/isthmusNode06|serverindex.xml#ServerIndex_1
```

4. Modify the host name for the application server, similar to the following example:

Enter the following line of code:

```
AdminConfig.modify('(cells/isthmusCell116/nodes/isthmusNode06|serverindex.xml  
#ServerIndex_1)', "[[hostName new_host_name]]")
```

The command is split on multiple lines for printing purposes.

5. Modify the host name for the Daemon instance, similar to the following example:

Enter the following line of code:

```
AdminTask.modifyNodeGroupProperty('DefaultNodeGroup', '[ -name was.WAS_DAEMON_protocol_iiop_daemon_listenIPAddress -value newHostname]')
```

6. Verify that the host name is correct, similar to the following example:

Enter the following line of code:

```
AdminConfig.show('(cells/isthmusCell107/nodes/isthmusCellManager07|  
serverindex.xml#ServerIndex_1)', 'hostName')
```

The response is:

```
'[hostName isthmus]'
```

The command is split on multiple lines for printing purposes.

7. Save the configuration.

Enter the following line of code:

```
AdminConfig.save()
```

8. Type `exit` to end the `wsadmin` session.

9. Update the application server with the changes.

- a. Stop the application server.

Enter the following command:

```
stopServer server1 -profileName AppSrv01
```

- b. Restart the application server.

Enter the following command:

```
startServer server1 -profileName AppSrv01
```

- Correct the host names for the ports that an application server opens.

If you have to correct the host names of the server ports, then you can make the correction using command line tools and either the `wsadmin` scripting tool or the administrative console. You might have to correct the host names of multiple ports for a particular server. This example shows you how to correct the host names using the administrative console and command line tools.

1. For the application server, select **Servers > Server Types > WebSphere application servers > application_server > Ports**.

2. Select a port whose host name needs changing.

3. Change the host name in the **Host** field; Click **OK**.

4. Continue selecting ports and changing host names until you correct each of the host names for the server ports.

5. Save the changes to the master configuration.

6. Update the application server with the changes.

- a. Stop the application server.

– Select **Servers > Server Types > WebSphere application servers**.

– Select the server that you want to stop.

– Click **Stop**.

- b. Restart the application server.

Enter the following command:

```
startServer server1 -profileName AppSrv01
```

Results

You have changed the host name of the server, the host names of the server ports, or both.

What to do next

You can continue to administer the product by doing tasks such as deploying the applications that you want to run on this server.

Administering stand-alone nodes using the administrative agent

You can configure an administrative agent and view or change stand-alone application server nodes registered to the administrative agent. An administrative agent provides a single interface to administer application servers in, for example, development, unit test, or server farm environments.

Before you begin

Install the WebSphere Application Server product.

About this task

The administrative agent provides a single interface to administer multiple stand-alone application server nodes in, development, unit test, or server farm environments, for example. By using a single interface to administer your application servers, you reduce the overhead of running administrative services in every application server.

You can use the administrative console of the administrative agent to configure the administrative agent and view and change properties for nodes registered to the administrative agent.

Procedure

- Set up the administrative agent environment.
 - Create an administrative agent profile and one or more stand-alone application server profiles, called *nodes*, on the same computer and then register the node profiles with the administrative agent.
- Start and stop the administrative agent as needed.
- View and change properties for the administrative agent.
 1. Click **System Administration** > **Administrative agent** from the navigation of the administrative agent administrative console.
 - Optionally view the administrative agent properties on the Configuration tab and the Runtime tab.
 - Optionally select **Start components as needed** on the Configuration tab. Click **Apply**, and then click **OK**.

Selecting the setting allows administrative agent components to start dynamically as needed for applications.
- View and change properties for a node registered to the administrative agent.
 1. Click **System Administration** > **Administrative agent** > **Nodes**.
 - You can view the nodes registered to the administrative agent.
 2. Click **System Administration** > **Administrative agent** > **Nodes** > *node_name*.

The page is read-only. To change properties for the node, click the links under Additional Properties.
- Unregister a stand-alone application server node from the administrative agent.

Unregister nodes if you no longer need the node in the administrative agent environment or if you intend to delete the stand-alone application server node profile. Run the `deregisterNode` command to unregister a node.

Results

Depending on the steps that you completed, you might have configured the administrative agent, or viewed or changed properties for a node registered to the administrative agent.

What to do next

You can continue to administer registered nodes from the administrative agent. You can further configure the administrative agent using the links on the configuration tab of the administrative agent panel. You can register more nodes with the administrative agent using the `registerNode` command. You can unregister nodes from the administrative agent using the `deregisterNode` command.

Administrative agent

An administrative agent provides a single interface to administer multiple application servers with stand-alone nodes in environments such as development, unit test, or that portion of a server farm that resides on a single machine.

The administrative agent and application servers must be on the same machine, but you can connect to the machine from a browser or the `wsadmin` tool on another machine.

gotcha: Registered nodes must have the same products as the administrative agent, and the products must be at the same version levels on the registered node and the administrative agent. This requirement is enforced because the administrative agent must have a matching environment in order to handle all of the administrative capabilities of the registered node. A node is not allowed to register with an administrative agent unless that node has an identical set of products and versions.

transition: If you were previously running on Version 7.0.0.11 or earlier, and have an administrative agent with a managed node that has mismatched products or versions, when you when you migrate to Version 8.0, that administrative agent will not be able to start the subsystem for any mismatched nodes. You must update these nodes to have the same products and versions as the administrative agents, restart the servers on the node and then restart the administrative agent, before the administrative agent can resume managing these registered nodes

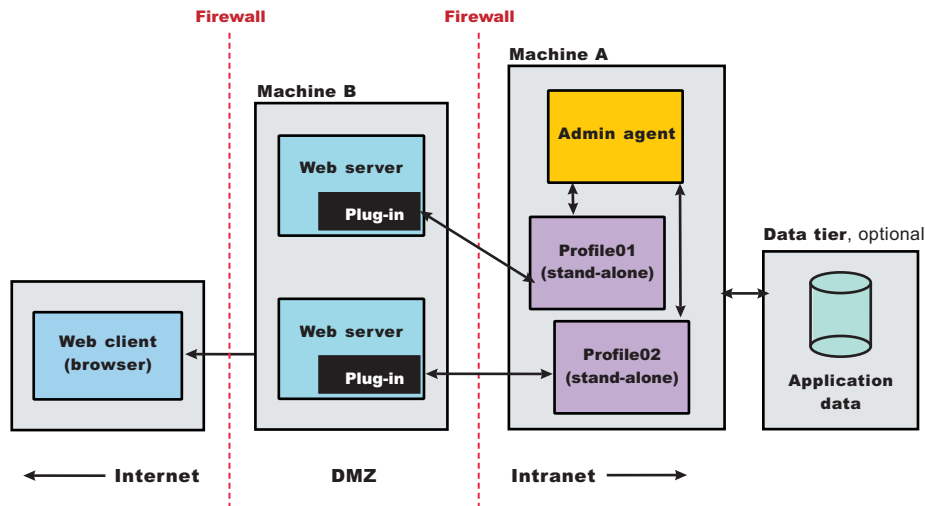
An administrative agent can monitor and control multiple application servers on one or more nodes. Use the application servers only to run your applications. By using a single interface to administer your application servers, you reduce the overhead of running administrative services in every application server.

You can use the administrative agent to install applications on application servers, change application server configurations, stop and restart application servers, and create additional application servers.

Example topology of multiple application servers managed by an administrative agent

The following example topology shows machine A with an administrative agent and two application servers, Profile01 and Profile02, that are registered with the administrative agent. The application servers on machine A each communicate with a remote web server on machine B through the web server plug-in. Firewalls provide additional security for the machines. Read the topic on planning to install WebSphere

Application Server for further information on the topology.



Administrative agent security

In a flexible management environment, a user ID must have the required authorization to use the administrative agent and to work with registered nodes.

Required security roles

You need the following roles to use the administrative agent:

Table 3. Required security roles for administrative agent tasks. Roles include administrator and roles required for the operation or node.

Administrative tasks	Required security roles
Register or unregister a base (stand-alone) node with the administrative agent	administrator
Work with the administrative agent:	Administrative roles required for the operation being performed
Work with the administrative subsystem, such as registered nodes	Administrative roles required for the registered base node

Same security domain configuration

The administrative agent supports a security configuration where all the cells in the topology share the same user registry, and therefore, the same security domain.

For the administrative agent topology, when a user logs in to the JMX connector port of an administrative subsystem, or chooses the registered node from the administrative console, the authorization table for the chosen node is used.

For example, suppose two stand-alone application servers, Node1 and Node2, are registered with an administrative agent. User1 is authorized as administrator for Node1, but is not authorized for Node2. User2 is authorized as configurator for Node2, but is not authorized for Node1. User1 can administer,

operate and configure Node1 and its resources. User2 can monitor and configure Node2 and its resources. Only User1 can register or unregister a node, Node1, with the administrative agent.

Do not use DMZ proxy

A DMZ proxy does not work with the administrative agent when security is enabled. Keep security enabled and do not use the administrative agent in a DMZ proxy environment.

Setting up the administrative agent environment

An administrative agent environment consists of an administrative agent and the stand-alone application servers that it manages. Setting up an administrative agent environment involves creating an administrative agent profile and one or more stand-alone application server profiles, called *nodes*, on the same computer and then registering the node profiles with the administrative agent.

Before you begin

Install the WebSphere Application Server product.

Make sure that the nodes that you want the administrative agent to manage have the same products as the administrative agent, and the products are at the same version levels on these nodes and the administrative agent. This requirement is enforced because the administrative agent must have a matching environment to handle all the administrative capabilities of the registered node. A node cannot register with an administrative agent unless that node has an identical set of products and versions.

A DMZ proxy does not work with the administrative agent when security is enabled. Keep security enabled and do not use the administrative agent in a DMZ proxy environment.

transition: If you were previously running on Version 7.0.0.11 or earlier, and have an administrative agent with a managed node that has mismatched products or versions, when you migrate to Version 8.0, that administrative agent will not be able to start the subsystem for any mismatched nodes. You must update these nodes to have the same products and versions as the administrative agents, restart the servers on the node and then restart the administrative agent, before the administrative agent can resume managing these registered nodes.

About this task

You can use an administrative agent to manage base (stand-alone) application servers that are on the same computer.

Administrative agents and the managed nodes are part of the flexible management environment.

To add an administrative agent to your environment, create an administrative agent profile using the `manageprofiles` command or the Profile Management Tool. To add a node, create a stand-alone application server profile and then register the stand-alone application server with the administrative agent.

The node must be on the same computer as the administrative agent.

Ensure that the profiles in the flexible management environment either all have security enabled or all have security disabled.

Procedure

1. Determine the topology for your administrative agent environment.
Determine which computers, stand-alone application server nodes, and node resources such as applications that you want to use.

To manage stand-alone application servers, use an administrative agent on each computer where the stand-alone application servers reside. For more information, see Scenarios 5 in the Planning to install WebSphere Application Server topic.

2. Determine the security roles needed for your administrative agent environment.

For an administrative agent environment, you typically have one administrative agent profile and one or more stand-alone application server profiles on the same computer. The stand-alone application server nodes are registered to the administrative agent. Profiles in the environment must either all have security enabled or all have security disabled. When you create the profiles, you can specify security options, user names, and passwords.

You must have security roles that authorize you to work with an administrative agent and to manage registered nodes and resources on those nodes. For more information, see the administrative agent security topic.

3. Create a management profile for the administrative agent.

You can use the Profile Management Tool or the `manageprofiles` command.

For example, in the Profile Management Tool, select the **Management** environment and click **Next**, select the **Administrative agent** server type, and select options that create the profile. By default, an administrative agent has its own administrative console, administrative security is enabled, and the console port is 9065. To disable administrative security, to specify a security certificate, or to change the default ports, use the advanced profile creation option when creating the administrative agent profile.

By default, the first administrative agent profile in a product installation is named `AdminAgent01` and its server name is `adminagent`.

For more information, see the topic on creating management profiles for administrative agents.

For `manageprofiles` examples, see the topic on the `manageprofiles` command. For `-templatePath`, specify the management template. For `-serverType`, specify `ADMIN_AGENT`.

4. Create profiles for the stand-alone application server nodes that you intend to have in your flexible management environment.

Create profiles for one or more stand-alone application server nodes that reside on the same computer as the administrative agent profile. You can use the Profile Management Tool or the `manageprofiles` command.

For example, in the Profile Management Tool, select the **Application server** environment and click **Next**, and then select options that create the profile. By default, an application server has its own administrative console, administrative security is enabled, and the console port is 9060. To disable administrative security, to specify a security certificate, to specify to install sample application, or to change the default ports, select the advanced profile creation option when creating the application server profile.

By default, the first application server profile in a product installation is named `AppSrv01` and its server name is `server1`.

For more information, see the topic on creating application server profiles.

For `manageprofiles` examples, see the topic on the `manageprofiles` command. For `-templatePath`, specify the default template. Do not specify a `-serverType` parameter.

5. Start the administrative agent server.

- Run the `startServer` command.

For example, suppose the `AdminAgent01` profile has the server name `adminagent`. Run the following command from the `bin` directory of the `AdminAgent01` profile:

```
startServer adminagent
```

If the administrative agent starts successfully, the open for e-business message displays and is written to the administrative agent `startServer.log` file:

```
Server launched. Waiting for initialization status.  
Server adminagent open for e-business; process id is 1932.
```

For more information, see the topic on starting and stopping the administrative agent.

6. Register the stand-alone application server nodes with the administrative agent.

Run the registerNode command of the administrative agent.

When you run the registerNode command, you can optionally specify parameters such as -node to assign a node name and -port to assign an administrative agent connector port. If security is enabled for the node that you are registering and the node user name and node password are different than those used for the administrative agent, specify values for -nodeusername and -nodepassword. For more information, see the topic on the registerNode command.

To register the AppSrv01 profile with the administrative agent, run the following command from the bin directory of the administrative agent profile:

```
registerNode -profilePath user_data_root/profiles/AppSrv01
```

For more information, see the topic on the registerNode command.

7. Verify that the nodes have been registered to the administrative agent.

You can use the administrative agent console or wsadmin scripting commands to see a list of nodes that are registered with the administrative agent.

- Use the administrative agent console to see a list of managed nodes.
 - a. Start the administrative agent console.
 - b. On the opening page of the administrative agent console, select to administer the administrative agent. The administrative agent has a name such as *host_nameAANode01*.
 - c. Log in to the administrative agent console.
 - d. Examine the Nodes page.
 - 1) Click **System administration > Administrative agent**.
 - 2) On the Configuration tab of the Administrative agent page, click **Nodes**.
 - e. Ensure that the Nodes page lists nodes that have been registered with the administrative agent.
- Use the AdminConfig list command to see a list of managed nodes. Run the following wsadmin scripting commands from the administrative agent bin directory.

- To use the Jython scripting language, enter the following two commands in succession:

```
wsadmin -lang jython  
  
print AdminConfig.list('ManagedNode')
```

- To use the Jacl scripting language, enter the following two commands in succession:

```
wsadmin  
  
$AdminConfig list ManagedNode
```

After you verify that the stand-alone application server nodes are registered with the administrative agent, enter quit to exit the wsadmin scripting tool.

8. Start the stand-alone application server nodes.

Run the startServer command.

```
startServer server1
```

If the server starts successfully, the open for e-business message displays and is written to the startServer.log file.

For more information, see topics on the startServer command and on starting application servers.

Results

The administrative agent environment is set up and the nodes are running.

What to do next

Use the administrative agent to monitor and configure the stand-alone application server nodes.

Starting and stopping the administrative agent

In your flexible management environment, you can start the administrative agent by using the `startServer` command. You can stop the administrative agent by using the `stopServer` command.

Before you begin

Before you can start or stop the administrative agent, you must first install the product.

About this task

Start the administrative agent so that you can manage multiple application servers. Stop the administrative agent as needed, such as when migrating to a new version of the product, when uninstalling the product, and so on.

Procedure

- Start the administrative agent.

Use one of these methods to start the administrative agent:

- Use the `startServer` command:

```
startServer <administrative_agent>
```

where *administrative_agent* is name of the administrative agent that you want to start.

Use the `startServer` Qshell script.

- Use the Submit Job (SBMJOB) CL command.

You can run the CL command from an IBM i command line:

```
SBMJOB CMD(CALL PGM(product_library/QWASSTRSVR) PARM('-profilePath'  
'profile_root' '-server' 'administrative_agent')) JOB(server)  
JOBQ(QWAS8/QWASJOBQ) JOBQ(QWAS8/QWASJOBQ) USER(QEJBSVR) LANGID(*USRPRF)  
CNTRYID(*USRPRF) CCSID(*USRPRF) OUTQ(QWAS8/QWASOUTQ) ALWMLTTHD(*YES)
```

- The *profile_root* element is the profile root of the administrative agent.
- The *administrative_agent* element is the name of the administrative agent server that you want to start.

- Stop the administrative agent.

Use one of these methods to stop the administrative agent:

- Use the `stopServer` command:

```
stopServer <administrative_agent>
```

where *administrative_agent* is name of the administrative agent that you want to stop.

Use the `stopServer` Qshell script.

- Use the end job (ENDJOB) CL command.

To use the ENDJOB CL command to end an administrative agent, enter this command on an IBM i command line:

```
ENDJOB JOB(jobNumber/QEJBSVR/jobName) OPTION(*CNTRLD) DELAY(delayTime)
```

The *jobNumber* element is the job number, the *jobName* element is the name of administrative agent job, and the *delayTime* element is the amount of time to wait for the job to end in seconds. Use a value of 600 seconds initially. Read about the appropriate delay time in the topic on shutting down the product subsystem.

Results

You have started the administrative agent and have optionally stopped it.

What to do next

Administer application servers using the administrative agent. You can do such tasks as configure the administrative agent, view or change properties for a node registered to the administrative agent, or view and change the job manager configuration for a registered node.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

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 products or components require multiple locations.

app_client_root

The default installation root directory for the Application Client for IBM WebSphere Application Server is the /QIBM/ProdData/WebSphere/AppClient/V8/client directory.

app_client_user_data_root

The default Application Client for IBM WebSphere Application Server user data root is the /QIBM/UserData/WebSphere/AppClient/V8/client directory.

app_client_profile_root

The default Application Client for IBM WebSphere Application Server profile root is the /QIBM/UserData/WebSphere/AppClient/V8/client/profiles/*profile_name* directory.

app_server_root

The default installation root directory for WebSphere Application Server is the /QIBM/ProdData/WebSphere/AppServer/V8/Base directory.

java_home

Table 4. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32-bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit
64-bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit

plugins_profile_root

The default Web Server Plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/*profile_name* directory.

plugins_root

The default installation root directory for Web Server Plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V8/webserver directory.

plugins_user_data_root

The default Web Server Plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver directory.

product_library

product_lib

This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS8x (where x is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed

on the system is QWAS8A. The *app_server_root/properties/product.properties* file contains the value for the product library of the installation, *was.install.library*, and is located under the *app_server_root* directory.

profile_root

The default directory for a profile named *profile_name* for WebSphere Application Server is the */QIBM/UserData/WebSphere/AppServer/V8/Base/profiles/profile_name* directory.

shared_product_library

The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root

The default user data directory for WebSphere Application Server is the */QIBM/UserData/WebSphere/AppServer/V8/Base* directory.

The *profiles* and *profileRegistry* subdirectories are created under this directory when you install the product.

web_server_root

The default web server path is */www/web_server_name*.

Administrative agent settings

Use this page to configure the administrative agent and view its properties.

To view this page in the administrative agent console, click **System administration > Administrative agent**.

Name

Specifies the administrative agent server name. The name is read-only.

Node

Specifies a name for the administrative agent node. The node name is unique within the cell. The node name is read-only.

By default, a node name is the hostname appended with Node01. For example, a node on a computer with the host name of MyComputer is named MyComputerNode01 by default.

However, the node name is a purely logical name for a group of servers. The node name does not have to contain the host name.

Start components as needed

Select this property if you want the server components started as they are needed for applications that run on this server.

When this property is not selected, all of the server components are started during the startup process. Therefore, selecting this property usually results in improved startup time because fewer components are started during the startup process.

gotcha: If you are running other WebSphere products on top of this product, make sure that those other products support this functionality before you select this property.

Process ID

Specifies the read-only process ID of the administrative agent.

Cell name

Specifies the read-only cell name of the administrative agent.

Node name

Specifies the read-only node name of the administrative agent.

State

Specifies the read-only state of the administrative agent, such as started or stopped.

Node collection for the administrative agent

Use this page to view the application server nodes that are registered to the administrative agent. The administrative agent provides a single interface to the registered nodes.

To view this administrative console page, click **System administration > Administrative agent > Nodes**.

Name

Specifies a name for an application server node that is registered to the administrative agent. The name is read-only.

Registered nodes settings

Use this page to view properties for a node registered to the administrative agent. The properties are name, unique ID, and poll jobs from job manager.

To view this administrative console page, click **System administration > Administrative agent > Nodes > *node_name***.

Name: Specifies the name of an application server registered to the administrative agent. The name is read-only.

Unregistering nodes of the administrative agent

You can unregister application server nodes so that they are no longer registered to an administrative agent. Unregister nodes if you no longer need the node in the administrative agent environment or if you intend to delete the application server node profile. After you unregister a node from an administrative agent, you can use the node stand-alone, register the node with another administrative agent, or delete the application server node profile.

Before you begin

The application server node that you want to remove from the administration agent environment must be registered with the administrative agent. Start the administrative agent if it is not running already.

About this task

To unregister a node, run the `deregisterNode` command from the `bin` directory of the administrative agent. Step 1 describes how to run the `deregisterNode` command.

When you unregister a node, the node configuration is retained, but is marked as not registered with the administrative agent. If the node that you unregister had the administrative console or management Enterprise JavaBeans (EJB) applications installed before registering the node, they are re-enabled.

Running the `deregisterNode` command might result in a null pointer exception if the application server node profile is corrupted or unusable. If you receive the null pointer exception, the process to unregister the application server from the administrative agent failed. You receive ADMU0116I, ADMU0128I, ADMU0211I, ADMU0113E, and ADMU1211I messages in the error log. Step 2 describes how to remove a node and related end points if there is a null pointer exception.

If the application server node profile is deleted before the node is unregistered, running the `deregisterNode` command is ineffective. Because the profile no longer exists, the administrative agent does not recognize

the profile. Complete Step 2 to remove the node and related end points from the administrative agent environment.

Procedure

1. Unregister a node using the `deregisterNode` command.

If the node that you want to unregister exists, run the `deregisterNode` command, specifying the profile path of the node to unregister:

```
deregisterNode -profilePath profile_root/profile_name
```

For example, to unregister the `AppSrv02` profile from the administrative agent environment, run the following command:

```
deregisterNode -profilePath profile_root/AppSrv02
```

See the topic on the `deregisterNode` command for information about command parameters.

2. If a null pointer exception results from running the `deregisterNode` command or if the node profile has been deleted, run `wsadmin` commands that remove the registered node and related end points.

- a. On a command line, run a command to start the `wsadmin` scripting tool from the administrative agent `bin` directory.

To use the Jython scripting language, enter:

```
wsadmin -lang jython
```

To use the Jacl scripting language, enter:

```
wsadmin
```

- b. If you do not know the name of the node to remove, run the `AdminConfig list` command to list nodes that are registered with the administrative agent and find the node to remove in the list.

For Jython:

```
print AdminConfig.list('ManagedNode')
```

For Jacl:

```
$AdminConfig list ManagedNode
```

The list of registered nodes that is displayed resembles the following:

```
nodeA(cells/myAACell101/managednodes/nodeA|managednode.xml#ManagedNode_1239121412703)
nodeB(cells/myAACell101/managednodes/nodeB|managednode.xml#ManagedNode_1239121498500)
```

This list shows that `nodeA` and `nodeB` are registered nodes of the `myAACell101` administrative agent.

- c. Issue `wsadmin` commands that remove the node.

To remove `nodeA` and save the changes, run the following commands in succession.

For Jython:

```
mn = AdminConfig.getid('/ManagedNode:nodeA/')
```

```
AdminConfig.remove(mn)
```

```
AdminConfig.save()
```

For Jacl:

```
set mn [$AdminConfig getid /ManagedNode:nodeA/]
```

```
$AdminConfig remove $mn
```

```
$AdminConfig save
```

- d. Run `wsadmin` commands that remove end points that were generated for the subsystem when the node profile was registered.

Run the following commands sequentially to remove end points for `nodeA`. The `for` command in Jython and the `foreach` command in Jacl are one-line commands that are shown on multiple lines for publication.

For Jython:


```

import java.lang.System as System

lineSeparator = System.getProperty("line.separator")

neps = AdminConfig.list("NamedEndPoint").split(lineSeparator)

for nep in neps:
    set name = AdminConfig.showAttribute(nep, "endPointName")
    if (name.endswith("nodeA") == 1):
        AdminConfig.remove(nep)

AdminConfig.save()

quit

```

For Jacl:

```

set neps [$AdminConfig list NamedEndPoint]

foreach nep $neps {set name [$AdminConfig showAttribute $nep endPointName];
if {[string last "nodeA" $name] != -1} {$AdminConfig remove $nep}}

$AdminConfig save

quit

```

- e. Restart the administrative agent.

To restart an administrative agent named `adminagent`, run the following commands from a command prompt at the `bin` directory of the administrative agent profile:

```
stopServer adminagent
```

```
startServer adminagent
```

- f. Verify that the node is no longer registered with the administrative agent.

Results

The application server node is no longer registered with the administrative agent.

What to do next

You can use the `unregistered node stand-alone` or register the node with another administrative agent. Optionally, use the `manageprofiles` command to delete the application server profile.

Configuring administration services

You can configure administration services such as remote file services, repository services, and Java Management Extensions (JMX) connectors.

Remote files services for file transfer and file synchronization

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 deployment manager and the nodes as well as between the `wsadmin` scripting process and either the deployment manager or the application server. 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 the topic on port number settings in WebSphere Application Server versions.

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.

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 > Server Types > WebSphere 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

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 > Server Types > WebSphere 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

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 and IPC connector properties in the ipc.client.props file.

A Java Management Extensions (JMX) connector can be a Remote Method Invocation (RMI) connector, a Simple Object Access Protocol (SOAP) connector, a JMX Remote application programming interface (JSR 160) Remote Method Invocation (JSR160RMI) connector, or an Inter-Process Communications (IPC) connector.

Note: You should eventually convert all of your RMI connectors to JSR160RMI connectors because support for the RMI connector is deprecated.

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. Read the application programming interfaces documentation to learn how to code the JMX connector properties for a custom Java administrative client program.

The JMX connectors that servers create use JMX connector properties that are accessible in the administrative console. The wsadmin tool and the Java administrative client use JMX connector properties in the soap.client.props, ipc.client.props, and sas.client.prop files.

For the administrative console, this topic specifies the coding of the particular setting or property. Coding of properties in the soap.client.props file and the ipc.client.props file that are specific to JMX connectors is specified. These SOAP properties begin with com.ibm.SOAP and the IPC properties begin with com.ibm.IPC. Other properties in the soap.client.props file and the ipc.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 and the ipc.client.props file, is specified.

To view the JMX connector custom properties administrative console page that goes with this topic, click **Servers > Server Types > WebSphere application servers > *server_name* > Administration > Administration services > JMX connectors > *connector_type* > Custom properties.**

SOAP connector properties

This section discusses the following JMX connector properties that pertain to SOAP connectors:

- Configuration URL
- Secure Sockets Layer (SSL) security
- Security context provider
- SOAP request timeout
- SSL alias

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:// <i>Path</i> /soap.client.props
Default	None

- A Java administrative client. Use the AdminClient.CONNECTOR_SOAP_CONFIG property.

Secure Sockets Layer (SSL) security

Use this property to enable SSL security between the 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	<code>com.ibm.SOAP.securityEnabled</code>
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	<code>securityEnabled</code>
Data type	Boolean
Default	False

- A Java administrative client. Use the `AdminClient.CONNECTOR_SECURITY_ENABLED` property.

Security context provider

This property indicates the Secure Sockets Layer (SSL) implementation to use between the application server and the SOAP client.

Set the property by using the `soap.client.props` file.

Property	<code>com.ibm.ssl.contextProvider</code>
Data type	String
Valid Values	IBMJSSE2
Default	IBMJSSE2

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	<code>com.ibm.SOAP.requestTimeout</code>
Data type	Integer
Range in seconds	0 to n

Default	If the property is zero (0), the request never times out. 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
Default	If the property is zero (0), the request never times out. 600

- A Java administrative client. The property is AdminClient.CONNECTOR_SOAP_REQUEST_TIMEOUT.

SSL alias

This property specifies the alias to use for an SSL configuration for client connections. The value of the alias is what you want it to be.

Set the property in the `soap.client.props` file.

Property	com.ibm.ssl.alias
Data type	String
Default	DefaultSSLSettings

IPC connector properties

This section discusses the following JMX connector properties that pertain to IPC connectors:

- Configuration URL
- IPC request timeout
- Secure Sockets Layer (SSL) security
- Security context provider
- SSL alias

Configuration URL

Specify the configuration URL property if you want a program to read IPC 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.IPC.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	<code>http://Path/ipc.client.props</code>
Default	None

- A Java administrative client. Use the AdminClient.CONNECTOR_IPC_CONFIG property.

IPC 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 IPC client can override the default. Components that use the `ipc.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 `ipc.client.props` file.

Property	<code>com.ibm.IPC.requestTimeout</code>
Data type	Integer
Range in seconds	0 to n
Default	If the property is zero (0), the request never times out. 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	<code>requestTimeout</code>
Data type	Integer
Range in seconds	0 to n
Default	If the property is zero (0), the request never times out. 600

- A Java administrative client. The property is `AdminClient.CONNECTOR_IPC_REQUEST_TIMEOUT`.

Secure Sockets Layer (SSL) security

Use this property to enable SSL security between Application Server and the IPC client. Set the property by using one of the following options:

- Scripts that run from a command-line interface.
- The `ipc.client.props` file.

Property	<code>com.ibm.IPC.securityEnabled</code>
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	<code>securityEnabled</code>
Data type	Boolean
Default	False

- A Java administrative client. Use the `AdminClient.CONNECTOR_SECURITY_ENABLED` property.

Security context provider

This property indicates the SSL implementation to use between the application server and the IPC client.

Set the property by using the `ipc.client.props` file.

Property	<code>com.ibm.ssl.contextProvider</code>
Data type	String
Valid Values	IBMJSSE2
Default	IBMJSSE2

SSL alias

This property specifies the alias to use for an SSL configuration for client connections. The value of the alias is what you want it to be.

Set the property in the `ipc.client.props` file.

Property	<code>com.ibm.ssl.alias</code>
Data type	String
Default	DefaultSSLSettings

SOAP, RMI, JSR160RMI, and IPC connector properties

This section discusses JMX connector properties that pertain to the following SOAP connectors, RMI connectors, JSR160RMI connectors, and IPC connectors:

- Connector type
- Disabling a connector
- Host
- Password
- Port
- User name

Connector type

A connector type of SOAP, RMI, JSR160RMI, or IPC depends on whether the application server connects to a SOAP server, an RMI server, a JSR160RMI server, or an IPC 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 JSR160RMIConnector IPCConnector		
Default	SOAPConnector	JSR160RMI	IPC

- A Java administrative client. Use the `AdminClient.CONNECTOR_TYPE` property. Specify the connector type by using the `AdminClient.CONNECTOR_TYPE_RMI`, the `AdminClient.CONNECTOR_TYPE_SOAP`, the `AdminClient.CONNECTOR_TYPE_JSR160RMI`, or the `AdminClient.CONNECTOR_TYPE_IPC` constants.

Disabling a connector

You can enable or disable any of the JMX connectors from the administrative console.

- The wsadmin tool.
- The administrative console. Select the box next to the connector to enable the connector. Clear the box next to the connector to disable the connector.

Property	enabled
Data type	Boolean
Value	true/false

Host

The host name or the IP address of the server to which the application server connects. The server can be a SOAP server, an RMI server, a JSR160RMI server, or an IPC 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.

Password

The password that the application server uses to access the SOAP server, the RMI server, the JSR160RMI server, or the IPC 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 for the SOAP server, the RMI server, or the JSR160RMI server.

Property	com.ibm.SOAP.loginPassword
Data type	String
Valid values	The value must match the global SSL settings for SOAP, RMI, or JSR160RMI.
Default	None

- The ipc.client.props file for the IPC server.

Property	com.ibm.IPC.loginPassword
Data type	String
Valid values	The value must match the global SSL settings for IPC.
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, RMI, JSR160RMI, or IPC.
Default	None

- A Java administrative client. Use the AdminClient.PASSWORD property.

Port

The port number of the server to which the application server connects. The server can be a SOAP server, an RMI server, a JSR160RMI server, or an IPC 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 the application server uses to access the SOAP server, an RMI server, a JSR160RMI server, or an IPC 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 for the SOAP server, an RMI server, a JSR160RMI server.

Property	com.ibm.SOAP.loginUserid
Data type	String
Valid value	The value must match the global SSL settings for SOAP, RMI, or JSR160RMI.
Default	None

- The ipc.client.props file for the IPC server.

Property	com.ibm.IPC.loginUserid
Data type	String
Valid value	The value must match the global SSL settings for IPC.
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, RMI, JSR160RMI, or IPC.
Default	None

- A Java administrative client. Use the AdminClient.USERNAME property.

RMI connector properties

This section discusses the following JMX connector properties that pertain to RMI connectors:

- Disabling the JSR 160 RMI connector

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 (JMX) connectors

Use this page to view and change the configuration for Java Management Extensions (JMX) connectors, which make connections between server processes. The types of JMX connectors are Simple Object Access Protocol (SOAP), Remote Method Invocation (RMI), JMX Remote application programming interface (JSR 160) Remote Method Invocation (JSR160RMI), and Inter-Process Communications (IPC).

To view this administrative console page, click **Servers > Server Types > WebSphere application 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 SOAP connector and an appropriate port number. You can also use the RMI connector, the JSR160RMI connector, or the IPC 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	Enumeration
Default	SOAPConnector
Range	<p>SOAPConnector For JMX connections using Simple Object Access Protocol (SOAP).</p> <p>RMIConnector For JMX connections using Remote Method Invocation (RMI).</p> <p>JSR160RMIConnector For JMX connections using JMX Remote application programming interface (JSR 160) Remote Method Invocation (JSR160RMI).</p> <p>IPCConnector For JMX connections using Inter-Process Communications (IPC).</p>

Enabled

Specifies whether a JMX connector is enabled. If Yes is specified, the connector is enabled. All JMX connectors are enabled by default.

To disable a JMX connector, select the connector and click **Disable**. The **Enabled** value changes to No. To enable a JMX connector, select the connector and click **Enable**. The Enabled value changes to Yes.

Data type	Boolean
------------------	---------

JMX connector settings

Use this page to view the configuration for a Java Management Extensions (JMX) connector, which makes connections between server processes.

To view this administrative console page, click **Servers > Server Types > WebSphere application servers > *server_name* > Administration > Administration services > JMX Connectors > *connector_type***.

Type:

Specifies the type of the JMX connector.

Data type
Default
Range

Enumeration
 SOAPConnector
SOAPConnector
 For JMX connections using Simple Object Access Protocol (SOAP).
RMICConnector
 For JMX connections using Remote Method Invocation (RMI).
JSR160RMICConnector
 For JMX connections using JMX Remote application programming interface (JSR 160) Remote Method Invocation (JSR160RMI).
IPCConnector
 For JMX connections using Inter-Process Communications (IPC).
gotcha: IPC_CONNECTOR_ADDRESS is a special end point. The end point must be defined as 'localhost'. Editing a hostname definition from 'localhost' to any other <hostname> is not permitted if you plan to use this same server as a template to create another server in another node and which is to have an end point definition different than <hostname>.

SOAP connector and Inter-Process Communications connector properties files

Use the soap.client.props file to set properties for the SOAP connector and the ipc.client.props file to set properties for the Inter-Process Communications (IPC) connector. Most of the properties in the ipc.client.props file have corresponding properties in the soap.client.props file.

The SOAP connector properties file for a particular profile is at the following location:

- *profile_root*/properties/soap.client.props

The IPC connector properties file for a particular profile is at the following location:

- *profile_root*/properties/ipc.client.props

The following table provides basic information on the various properties. Read the properties files to obtain more detailed information.

Table 5. SOAP connector and IPC connector property descriptions. The properties configure the SOAP and IPC connectors.

SOAP connector properties	IPC connector properties	Description
com.ibm.SOAP.securityEnabled	com.ibm.IPC.securityEnabled	Specifies enablement of security for the connector. Set the property to true to enable security.
com.ibm.SOAP.authenticationTarget	com.ibm.IPC.authenticationTarget	Specifies the type of authentication for the connector if security is enabled. You can specify BasicAuth for basic authentication. If no value is specified, basic authentication is used.

Table 5. SOAP connector and IPC connector property descriptions (continued). The properties configure the SOAP and IPC connectors.

SOAP connector properties	IPC connector properties	Description
com.ibm.SOAP.loginUserId	com.ibm.IPC.loginUserId	Specifies the user ID for the connector if security is enabled, and you do not enter a user ID through a command prompt or standard in.
com.ibm.SOAP.loginPassword	com.ibm.IPC.loginPassword	Specifies the password for the connector if security is enabled, and you do not enter a password through a command prompt or standard in.
com.ibm.SOAP.loginSource	com.ibm.IPC.loginSource	Specifies automatic prompting for the user ID and password when you specify prompt. Prerequisites for using this property are discussed in the properties file for the particular connector.
com.ibm.SOAP.requestTimeout	com.ibm.IPC.requestTimeout	Specifies how long in seconds the connector waits for a server response. The property for the SOAP connector and the property for the IPC connector are each initially set to 180 in their respective properties files.
com.ibm.ssl.alias	com.ibm.ssl.alias	This property specifies the alias to use for a Secure Sockets Layer (SSL) configuration for client connections. The value of the alias is what you want it to be.
	timeToExpiration	Specifies the time in seconds that connections can be idle in the connection pool. Beyond this time the connections are purged. The initial setting for the property is 360.

Note: Many of the system management commands contain implicit stop server operations. Most of these commands use the IPC connector properties configuration. However, some commands, such as the stopServer, stopNode, and stopManager commands, continue to use the SOAP connector properties configuration for compatibility reasons. Thus, to avoid additional user ID and password prompts, specify the user ID and password information in both the soap.client.props and ipc.client.props files.

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 > Server Types > WebSphere 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 > Server Types > WebSphere application servers > *server_name* > Administration > Administration services > Extension MBean Providers > *provider_library_name*.**

Name:

The name used to identify the Extension MBean provider library.

Data type String

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

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 > Server Types > WebSphere 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 > Server Types > WebSphere 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

Administrative audit messages in system logs

The product provides administrative audit messages in system logs that contain some audit information. The audit messages described in this topic are part of the standard product audit stream and do not provide administrative event auditing information such as who changed files.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Important: The functionality described in this topic uses system logs and is not a part of the security auditing subsystem. The audit information captured by this functionality does not correspond with the audit information captured by the security auditing subsystem. For information about the security auditing subsystem, see the topic on auditing the security infrastructure.

Administrative audits use the same trace logging facility as the rest of the product, and do not use the logging facility that is a part of the security auditing subsystem. 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.

You can do administrative audits with or without the security audit facility.

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 servers and applications. These managed bean (MBean) operations provide administrative auditing:

Table 6. Administrative auditing MBean operations. The MBean types provide administrative auditing MBean operations.

MBean type	MBean operations
Server	stop, stopImmediate

Configuration change audits have ADMRxxxxl message IDs, where xxxx is the message number. Operational audits have ADMN10xxl message IDs, where 10xx is the message number.

Here are some audit examples for the application server environment. The audit examples are found in the application server 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)
...
[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.

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 and IPC connector properties in the ipc.client.props file.

A Java Management Extensions (JMX) connector can be a Remote Method Invocation (RMI) connector, a Simple Object Access Protocol (SOAP) connector, a JMX Remote application programming interface (JSR 160) Remote Method Invocation (JSR160RMI) connector, or an Inter-Process Communications (IPC) connector.

Note: You should eventually convert all of your RMI connectors to JSR160RMI connectors because support for the RMI connector is deprecated.

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. Read the application programming interfaces documentation to learn how to code the JMX connector properties for a custom Java administrative client program.

The JMX connectors that servers create use JMX connector properties that are accessible in the administrative console. The wsadmin tool and the Java administrative client use JMX connector properties in the soap.client.props, ipc.client.props, and sas.client.prop files.

For the administrative console, this topic specifies the coding of the particular setting or property. Coding of properties in the soap.client.props file and the ipc.client.props file that are specific to JMX connectors is specified. These SOAP properties begin with com.ibm.SOAP and the IPC properties begin with com.ibm.IPC. Other properties in the soap.client.props file and the ipc.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 and the ipc.client.props file, is specified.

To view the JMX connector custom properties administrative console page that goes with this topic, click **Servers > Server Types > WebSphere application servers > *server_name* > Administration > Administration services > JMX connectors > *connector_type* > Custom properties.**

SOAP connector properties

This section discusses the following JMX connector properties that pertain to SOAP connectors:

- Configuration URL
- Secure Sockets Layer (SSL) security
- Security context provider
- SOAP request timeout
- SSL alias

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	<code>http://Path/soap.client.props</code>
Default	None

- A Java administrative client. Use the `AdminClient.CONNECTOR_SOAP_CONFIG` property.

Secure Sockets Layer (SSL) security

Use this property to enable SSL security between the 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	<code>com.ibm.SOAP.securityEnabled</code>
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	<code>securityEnabled</code>
Data type	Boolean
Default	False

- A Java administrative client. Use the `AdminClient.CONNECTOR_SECURITY_ENABLED` property.

Security context provider

This property indicates the Secure Sockets Layer (SSL) implementation to use between the application server and the SOAP client.

Set the property by using the `soap.client.props` file.

Property	<code>com.ibm.ssl.contextProvider</code>
Data type	String
Valid Values	IBMJSSE2
Default	IBMJSSE2

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	<code>com.ibm.SOAP.requestTimeout</code>
Data type	Integer
Range in seconds	0 to n
Default	If the property is zero (0), the request never times out. 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	<code>requestTimeout</code>
Data type	Integer
Range in seconds	0 to n
Default	If the property is zero (0), the request never times out. 600

- A Java administrative client. The property is `AdminClient.CONNECTOR_SOAP_REQUEST_TIMEOUT`.

SSL alias

This property specifies the alias to use for an SSL configuration for client connections. The value of the alias is what you want it to be.

Set the property in the `soap.client.props` file.

Property	<code>com.ibm.ssl.alias</code>
Data type	String
Default	<code>DefaultSSLSettings</code>

IPC connector properties

This section discusses the following JMX connector properties that pertain to IPC connectors:

- Configuration URL
- IPC request timeout
- Secure Sockets Layer (SSL) security
- Security context provider
- SSL alias

Configuration URL

Specify the configuration URL property if you want a program to read IPC 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.IPC.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/ipc.client.props
Default	None

- A Java administrative client. Use the AdminClient.CONNECTOR_IPC_CONFIG property.

IPC 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 IPC client can override the default. Components that use the `ipc.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 `ipc.client.props` file.

Property	com.ibm.IPC.requestTimeout
Data type	Integer
Range in seconds	0 to n
Default	If the property is zero (0), the request never times out. 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
Default	If the property is zero (0), the request never times out. 600

- A Java administrative client. The property is AdminClient.CONNECTOR_IPC_REQUEST_TIMEOUT.

Secure Sockets Layer (SSL) security

Use this property to enable SSL security between Application Server and the IPC client. Set the property by using one of the following options:

- Scripts that run from a command-line interface.
- The `ipc.client.props` file.

Property	com.ibm.IPC.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.

Security context provider

This property indicates the SSL implementation to use between the application server and the IPC client.

Set the property by using the `ipc.client.props` file.

Property	com.ibm.ssl.contextProvider
Data type	String
Valid Values	IBMJSSE2
Default	IBMJSSE2

SSL alias

This property specifies the alias to use for an SSL configuration for client connections. The value of the alias is what you want it to be.

Set the property in the `ipc.client.props` file.

Property	com.ibm.ssl.alias
Data type	String
Default	DefaultSSLSettings

SOAP, RMI, JSR160RMI, and IPC connector properties

This section discusses JMX connector properties that pertain to the following SOAP connectors, RMI connectors, JSR160RMI connectors, and IPC connectors:

- Connector type
- Disabling a connector
- Host
- Password
- Port
- User name

Connector type

A connector type of SOAP, RMI, JSR160RMI, or IPC depends on whether the application server connects to a SOAP server, an RMI server, a JSR160RMI server, or an IPC 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 JSR160RMIConnector IPCConnector		
Default	SOAPConnector	JSR160RMI	IPC

- A Java administrative client. Use the AdminClient.CONNECTOR_TYPE property. Specify the connector type by using the AdminClient.CONNECTOR_TYPE_RMI, the AdminClient.CONNECTOR_TYPE_SOAP, the AdminClient.CONNECTOR_TYPE_JSR160RMI, or the AdminClient.CONNECTOR_TYPE_IPC constants.

Disabling a connector

You can enable or disable any of the JMX connectors from the administrative console.

- The wsadmin tool.
- The administrative console. Select the box next to the connector to enable the connector. Clear the box next to the connector to disable the connector.

Property	enabled
Data type	Boolean
Value	true false

Host

The host name or the IP address of the server to which the application server connects. The server can be a SOAP server, an RMI server, a JSR160RMI server, or an IPC 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.

Password

The password that the application server uses to access the SOAP server, the RMI server, the JSR160RMI server, or the IPC 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 for the SOAP server, the RMI server, or the JSR160RMI server.

Property	com.ibm.SOAP.loginPassword
Data type	String
Valid values	The value must match the global SSL settings for SOAP, RMI, or JSR160RMI.
Default	None

- The ipc.client.props file for the IPC server.

Property	com.ibm.IPC.loginPassword
Data type	String
Valid values	The value must match the global SSL settings for IPC.
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, RMI, JSR160RMI, or IPC.
Default	None

- A Java administrative client. Use the AdminClient.PASSWORD property.

Port

The port number of the server to which the application server connects. The server can be a SOAP server, an RMI server, a JSR160RMI server, or an IPC 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 the application server uses to access the SOAP server, an RMI server, a JSR160RMI server, or an IPC 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 for the SOAP server, an RMI server, a JSR160RMI server.

Property	com.ibm.SOAP.loginUserid
Data type	String
Valid value	The value must match the global SSL settings for SOAP, RMI, or JSR160RMI.
Default	None

- The ipc.client.props file for the IPC server.

Property	com.ibm.IPC.loginUserid
Data type	String
Valid value	The value must match the global SSL settings for IPC.
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, RMI, JSR160RMI, or IPC.
Default	None

- A Java administrative client. Use the AdminClient.USERNAME property.

RMI connector properties

This section discusses the following JMX connector properties that pertain to RMI connectors:

- Disabling the JSR 160 RMI connector

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 (JMX) connectors

Use this page to view and change the configuration for Java Management Extensions (JMX) connectors, which make connections between server processes. The types of JMX connectors are Simple Object Access Protocol (SOAP), Remote Method Invocation (RMI), JMX Remote application programming interface (JSR 160) Remote Method Invocation (JSR160RMI), and Inter-Process Communications (IPC).

To view this administrative console page, click **Servers > Server Types > WebSphere application 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 SOAP connector and an appropriate port number. You can also use the RMI connector, the JSR160RMI connector, or the IPC 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	Enumeration
Default	SOAPConnector

Range	<p>SOAPConnector For JMX connections using Simple Object Access Protocol (SOAP).</p> <p>RMIConnector For JMX connections using Remote Method Invocation (RMI).</p> <p>JSR160RMIConnector For JMX connections using JMX Remote application programming interface (JSR 160) Remote Method Invocation (JSR160RMI).</p> <p>IPCConnector For JMX connections using Inter-Process Communications (IPC).</p>
--------------	---

Enabled

Specifies whether a JMX connector is enabled. If Yes is specified, the connector is enabled. All JMX connectors are enabled by default.

To disable a JMX connector, select the connector and click **Disable**. The **Enabled** value changes to No. To enable a JMX connector, select the connector and click **Enable**. The Enabled value changes to Yes.

Data type Boolean

JMX connector settings

Use this page to view the configuration for a Java Management Extensions (JMX) connector, which makes connections between server processes.

To view this administrative console page, click **Servers > Server Types > WebSphere application servers > server_name > Administration > Administration services > JMX Connectors > connector_type**.

Type:

Specifies the type of the JMX connector.

Data type	Enumeration
Default	SOAPConnector
Range	<p>SOAPConnector For JMX connections using Simple Object Access Protocol (SOAP).</p> <p>RMIConnector For JMX connections using Remote Method Invocation (RMI).</p> <p>JSR160RMIConnector For JMX connections using JMX Remote application programming interface (JSR 160) Remote Method Invocation (JSR160RMI).</p> <p>IPCConnector For JMX connections using Inter-Process Communications (IPC). gotcha: IPC_CONNECTOR_ADDRESS is a special end point. The end point must be defined as 'localhost'. Editing a hostname definition from 'localhost' to any other <hostname> is not permitted if you plan to use this same server as a template to create another server in another node and which is to have an end point definition different than <hostname>.</p>

SOAP connector and Inter-Process Communications connector properties files

Use the `soap.client.props` file to set properties for the SOAP connector and the `ipc.client.props` file to set properties for the Inter-Process Communications (IPC) connector. Most of the properties in the `ipc.client.props` file have corresponding properties in the `soap.client.props` file.

The SOAP connector properties file for a particular profile is at the following location:

- `profile_root/properties/soap.client.props`

The IPC connector properties file for a particular profile is at the following location:

- `profile_root/properties/ipc.client.props`

The following table provides basic information on the various properties. Read the properties files to obtain more detailed information.

Table 7. SOAP connector and IPC connector property descriptions. The properties configure the SOAP and IPC connectors.

SOAP connector properties	IPC connector properties	Description
<code>com.ibm.SOAP.securityEnabled</code>	<code>com.ibm.IPC.securityEnabled</code>	Specifies enablement of security for the connector. Set the property to <code>true</code> to enable security.
<code>com.ibm.SOAP.authenticationTarget</code>	<code>com.ibm.IPC.authenticationTarget</code>	Specifies the type of authentication for the connector if security is enabled. You can specify <code>BasicAuth</code> for basic authentication. If no value is specified, basic authentication is used.
<code>com.ibm.SOAP.loginUserId</code>	<code>com.ibm.IPC.loginUserId</code>	Specifies the user ID for the connector if security is enabled, and you do not enter a user ID through a command prompt or standard in.
<code>com.ibm.SOAP.loginPassword</code>	<code>com.ibm.IPC.loginPassword</code>	Specifies the password for the connector if security is enabled, and you do not enter a password through a command prompt or standard in.
<code>com.ibm.SOAP.loginSource</code>	<code>com.ibm.IPC.loginSource</code>	Specifies automatic prompting for the user ID and password when you specify prompt. Prerequisites for using this property are discussed in the properties file for the particular connector.
<code>com.ibm.SOAP.requestTimeout</code>	<code>com.ibm.IPC.requestTimeout</code>	Specifies how long in seconds the connector waits for a server response. The property for the SOAP connector and the property for the IPC connector are each initially set to 180 in their respective properties files.
<code>com.ibm.ssl.alias</code>	<code>com.ibm.ssl.alias</code>	This property specifies the alias to use for a Secure Sockets Layer (SSL) configuration for client connections. The value of the alias is what you want it to be.
	<code>timeToExpiration</code>	Specifies the time in seconds that connections can be idle in the connection pool. Beyond this time the connections are purged. The initial setting for the property is 360.

Note: Many of the system management commands contain implicit stop server operations. Most of these commands use the IPC connector properties configuration. However, some commands, such as the `stopServer`, `stopNode`, and `stopManager` commands, continue to use the SOAP connector properties configuration for compatibility reasons. Thus, to avoid additional user ID and password prompts, specify the user ID and password information in both the `soap.client.props` and `ipc.client.props` files.

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 > Server Types > WebSphere 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 > Server Types > WebSphere application servers > *server_name* > Administration > Administration services > Extension MBean Providers > *provider_library_name***.

Name:

The name used to identify the Extension MBean provider library.

Data type String

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

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 > Server Types > WebSphere 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 > Server Types > WebSphere 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

Administrative audit messages in system logs

The product provides administrative audit messages in system logs that contain some audit information. The audit messages described in this topic are part of the standard product audit stream and do not provide administrative event auditing information such as who changed files.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Important: The functionality described in this topic uses system logs and is not a part of the security auditing subsystem. The audit information captured by this functionality does not correspond with the audit information captured by the security auditing subsystem. For information about the security auditing subsystem, see the topic on auditing the security infrastructure.

Administrative audits use the same trace logging facility as the rest of the product, and do not use the logging facility that is a part of the security auditing subsystem. 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.

You can do administrative audits with or without the security audit facility.

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 servers and applications. These managed bean (MBean) operations provide administrative auditing:

Table 8. Administrative auditing MBean operations. The MBean types provide administrative auditing MBean operations.

MBean type	MBean operations
Server	stop, stopImmediate

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 audit examples for the application server environment. The audit examples are found in the application server 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)
...
[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.

Administration service settings

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

To view this administrative console page, click **Servers > Server Types > WebSphere application servers > server_name > Administration > Administration services**.

Remote connector

Specifies the remote JMX Connector type. The remote JMX connector is the connector that is used between server processes that reside on different physical machines, for example, between the deployment manager and the node agent. Available options of SOAPConnector, RMIConnector, and JSR160RMI Connector are defined using the JMX Connectors page.

Data type	String
Default	SOAPConnector

Local connector

Specifies the local JMX Connector type. The local JMX connector is the connector used between server processes that reside on the same physical machine, for example, between the node agent and its application servers. Available options of SOAPConnector, RMIConnector, JSR160RMI Connector, and IPC Connector are defined using the JMX Connectors page.

Data type	String
Default	IPC Connector

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 > Server Types > WebSphere application servers > *server_name* > Administration > Administration services > Custom properties**.

Specify a property and its value as a name-value pair on the Administration services custom properties page. You can use the custom properties page to define the following administration services custom properties:

- “com.ibm.websphere.mbeans.disableRouting”

com.ibm.websphere.mbeans.disableRouting

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. To turn off the default behavior, set the following custom property on the application server.

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.

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 both MBeans.
- `<on>WebSphere:type=custom,name=custommbean1</on><on>WebSphere:type=custom,name=custommbean2</on>`
The value disables the object name modification for both MBeans.

Administrative topology: Resources for learning

Use the following links to find relevant supplemental information about WebSphere Application Server administrative topologies 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 product, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and IBM 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

The site contains a listing of all WebSphere Application Server Redbooks.

- IBM WebSphere developerWorks

The 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, 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.ibm.com/software/support/> and search by product category or by product name. For example, if you are experiencing problems specific to the product, click **WebSphere Application Server** in the product list. The Support page is displayed.

Chapter 4. Working with server configuration files

This topic shows how to manage application server configuration files.

About this task

Application server configuration files define the available application servers, their configurations, and their contents.

A configuration repository stores configuration data.

By default, configuration repositories reside in the *config* subdirectory of the profile root directory.

Note: Do not store any non-default XML or XML backup files under the *profile_root/config* directory. Limit the *config* directory and subdirectories to valid default WebSphere Application Server configuration files. A variety of symptoms and exceptions are possible if non-default files are stored in this directory.

You should periodically save changes to your administrative configuration. You can change the default locations of configuration files, as needed.

Procedure

- 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.
 - Programming. 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. 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 in several documents in a cascading hierarchy of directories. Most configuration documents have XML content.

The configuration documents describe your server, its configuration, and its contents.

- “Hierarchy of directories of documents” on page 68
- “Changing configuration documents” on page 69

- “Transformation of configuration files” on page 69

Hierarchy of directories of documents

Changes made to the configuration documents are stored in the cell repository.

At the top of the hierarchy is the **cells** directory. It holds a subdirectory for the cell. The name of the cell subdirectory matches the name of the cell. For example, a cell named *cell1* has its configuration documents in the subdirectory *cell1*.

The subdirectories under the cell contain the entire set of documents for the node and server 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 to the node in the cell
- The **nodes** subdirectory, which holds a subdirectory for the node in the cell. The names of the nodes subdirectory matches the name of the node.

The 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, the server on the node uses the variable definition in the node document and ignores the definition in the cell document.

The node subdirectory holds a subdirectory for the server defined on the node. The name of the subdirectory matches the name of the server. The server subdirectory holds a `server.xml` file, which provides configuration data specific to the 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 Java 2 Platform, Enterprise Edition (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
    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.

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

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/ <i>cell_name/</i>	Define a role for administrative operation authorization.	
app.policy	config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i>	Define security permissions for application code.	X
cell.xml	config/cells/ <i>cell_name/</i>	Identify a cell.	
deployment.xml	config/cells/ <i>cell_name/</i> applications/ <i>application_name/</i>	Configure application deployment settings such as target servers and application-specific server configuration.	

filter.policy	config/cells/ <i>cell_name/</i>	Specify security permissions to be filtered out of other policy files.	X
integral-jms-authorizations.xml	config/cells/ <i>cell_name/</i>	Provide security configuration data for the integrated messaging system.	X
library.policy	config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i>	Define security permissions for shared library code.	X
multibroker.xml	config/cells/ <i>cell_name/</i>	Configure a data replication message broker.	
namestore.xml	config/cells/ <i>cell_name/</i>	Provide persistent name binding data.	X
naming-authz.xml	config/cells/ <i>cell_name/</i>	Define roles for a naming operation authorization.	X
node.xml	config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i>	Identify a node.	
pmirm.xml	config/cells/ <i>cell_name/</i>	Configure PMI request metrics.	X
resources.xml	config/cells/ <i>cell_name/</i> config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i> config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i> servers/ <i>server_name/</i>	Define operating environment resources, including JDBC, JMS, JavaMail, URL, JCA resource providers and factories.	
security.xml	config/cells/ <i>cell_name/</i>	Configure security, including all user ID and password data.	
server.xml	config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i> servers/ <i>server_name/</i>	Identify a server and its components.	
serverindex.xml	config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i>	Specify communication ports used on a specific node.	
spi.policy	config/cells/ <i>cell_name/</i> nodes/ <i>node_name/</i>	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
~	Tilde
(Left parenthesis
)	Right parenthesis

gotcha:

- Character restrictions are not enforced for DataSource, ServiceLog, GroupExt, UserExt, or SubjectExt object names.
- You can use one of the following methods to turn off character validation for custom property names, and for the name value of Property and J2EEResourceProperty configuration objects in wsadmin commands.

- Set the `com.ibm.websphere.management.configservice.validatePropNames` Java system property to `false` in the Java virtual machine (JVM) for the deployment manager server.
- Set the `com.ibm.websphere.management.configservice.validatePropNames` property using the **-javaoption** parameter when you use the `wsadmin` tooling in the local mode.

```
wsadmin -conntype none -javaoption  
"-Dcom.ibm.websphere.management.configservice.validatePropNames=false"
```

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.

Before you begin

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.

About this task

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:

Procedure

- 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.

Results

You loaded the saved configuration file if you chose to do so.

What to do next

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.

About this task

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:

Procedure

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.

About this task

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:

Procedure

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 the wstemp temporary workspace directory

Configuration changes are stored in the `wstemp` temporary workspace directory until the changes are merged with the master configuration repository. This topic discusses how to change the location of the `wstemp` temporary workspace directory.

Before you begin

You must first install WebSphere Application Server before you change the location of the `wstemp` directory, which is a temporary workspace directory.

About this task

Whenever a user logs into the administrative console, or uses `wsadmin` scripting to make a configuration change, the changes are stored in the workspace. When a user uses the ConfigService configuration service interface of the Java application programming interfaces (APIs), the user specifies a session object that is associated with the workspace in order to store the changes. Only when the user performs a save operation under the administrative console, `wsadmin` scripting, or the Java APIs are the changes propagated and merged with the master configuration repository. For each administrative console user or each invocation of `wsadmin` scripting, the application server creates a separate workspace directory to store the intermediate changes until the changes are merged with the master configuration repository. Users of the Java APIs use different session objects to decide where the workspace directory resides. Both the administrative console and `wsadmin` scripting generate user IDs randomly. The user IDs are different from the user IDs that you use to log into the administrative console or `wsadmin` scripting. The Java APIs can either randomly generate the user ID or specify the user ID as an option when creating the session object.

You might want to change the location of the `wstemp` directory if you want to keep it in a separate place from the product installation.

The product determines the location of the workspace in the following order by using the first Java virtual machine (JVM) property in the list that is set. If no JVM property is set, the product uses the default workspace location.

Table 9. Workspace locations of JVM system properties. The Location column states the `wstemp` directory location for specified JVM system properties.

JVM system property	Location	Comments
<code>websphere.workspace.root</code>	The <code>wstemp</code> directory location is the value of the <code>websphere.workspace.root</code> JVM system property plus <ul style="list-style-type: none"><code>/wstemp</code> For example, the <code>websphere.workspace.root</code> JVM system property and its value could be <ul style="list-style-type: none"><code>-Dwebsphere.workspace.root=/temp</code> The property and its value are split on multiple lines for printing purposes.	Set the JVM system property for the application server to change the <code>wstemp</code> directory location. Use the full path rather than a relative path for this property.
If the <code>websphere.workspace.root</code> property is not set, the value of the <code>user.install.root</code> property is used.	The default <code>wstemp</code> location is the value of the <code>user.install.root</code> JVM system property plus <ul style="list-style-type: none"><code>/wstemp</code>	Do not change the <code>user.install.root</code> property as the profile creation process sets this property by pointing to the <code>profile_root</code> directory. In this case, the <code>wstemp</code> location is: <ul style="list-style-type: none"><code>profile_root/wstemp</code>

Procedure

- Change the workspace location for a particular JVM property by setting the `-D` option on the `java` command.

This method of changing the workspace location is only needed when you run a stand-alone administrative program in local mode.

For example, use the following option:

`-Dwebsphere.workspace.root=the location of the new workspace directory`

- Change the JVM custom property through the administrative console by setting the JVM property as a name-value pair on the Custom properties page.

For example,

1. Click **Servers > Server Types > WebSphere application servers > *server_name* > Java and Process Management > Process definition > Java Virtual Machine > Custom properties.**
2. Click **New.**
3. Specify `websphere.workspace.root` as the name.
4. Specify the full path of the new workspace directory as the value. The `wstemp` directory is created under that path.
5. Stop the server.
This step is optional if you want to keep your existing workspace files.
6. Copy files from the old location of the workspace directory to the new location of the workspace directory.
This step is optional if you want to keep your existing workspace files.
7. Start the server.
This step is optional if you want to keep your existing workspace files.

Results

You have used either the administrative console or the `-D` option on the `java` command to change the location of the `wstemp` temporary workspace directory.

Backing up and restoring administrative configuration files

You can back up administrative configuration files using the `backupConfig` command. You can restore administrative configuration files using the `restoreConfig` command.

About this task

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.

Procedure

1. Run the `backupConfig` command to back up configuration files. See the `backupConfig` command topic in the *Using the administrative clients* PDF for information.
2. 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.

Before you begin

This topic assumes familiarity with the SAV and RST commands.

About this task

The administrative configuration contains vital information regarding your WebSphere Application Server setup. Back up configuration files on a regular basis.

Procedure

- 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/*')
```

Results

You have saved and backed up your WebSphere Application Server configuration.

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 IBM 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.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.

Configuration problem settings

Use this page to identify and view problems that exist in the current configuration.

To view this administrative console page, click **Troubleshooting > Configuration Problems** in the console navigation tree.

To view a configuration problem, click **Configuration Validation** in the console navigation tree, then select the type of configuration you want to view.

Configuration document validation

Use these fields to specify the level of validation to perform on configuration documents.

Maximum

Selecting **Maximum: Validate all documents** turns on validation for all documents, regardless of whether or not they are extracted, and regardless of the relationships between the documents.

High Selecting **High: Validate extracted, parent, and local sibling documents** turns on validation for extracted documents and their parent documents, and turns on validation for the sibling documents of the documents which have been extracted. For example, if **High** validation is selected, and if the `server.xml` document is extracted, when performing validation, validation is performed on the three documents: `server.xml`, `node.xml`, and `cell.xml`. and on the two sibling documents `variables.xml` and `resources.xml` within the `server1` directory.

Medium

Selecting **Medium: Validate extracted and parent documents** turns on validation for the documents which have been extracted by the user interface, and also turns on validation of the parent documents of the documents which have been extracted. For example, using the partial directory structure from above, if **Medium** validation is selected, and if the `server.xml` document is extracted, when performing validation, validation is performed on all three of the documents `server.xml`, `node.xml`, and `cell.xml`.

Low Selecting **Low: Validate extracted documents** turns on validation for just those documents which have been extracted by the user interface.

None Selecting **None: Do not validate documents** disables validation. No configuration documents are validated.

Enable Cross Validation

Enables cross validation of configuration documents. Enabling cross validation enables comparison of configuration documents for conflicting settings.

Configuration Problems

Displays current configuration problem error messages. Click a message for detailed information about the problem.

Scope

Sorts the configuration problem list by the configuration file where each error occurs. Click a message for detailed information about the problem.

Message

Displays the message returned from the validator.

Explanation

A brief explanation of the problem.

User action

Specifies the recommended action to correct the problem.

Target Object

Identifies the configuration object where the validation error occurred.

Severity

Indicates the severity of the configuration error. There are three possible values for severity.

Error This means that there is a problem with the configuration that might cause partial or complete failure of server function. This is the most severe warning.

Warning

This means that there is a problem with the configuration that might cause a failure of server function, or that might cause the server to function in an unexpected manner.

Information

A setting of the configuration that is unexpected and noteworthy, which requires customer notification. Information is used when the configuration has a value which is probably okay, but should be double checked by the administrator. This is the least crucial level of severity.

Local URI

Specifies the local URI of the configuration file where the error occurred.

Full URI

Specifies the full URI of the configuration file where the error occurred.

Validator classname

The classname of the validator reporting the problem.

Runtime events

Use the Runtime event pages of the administrative console to view the events published by application server classes.

To view these administrative console pages, click **Troubleshooting**. Expand **Runtime Messages** and click either **Runtime Error**, **Runtime Warning**, or **Runtime Information**.

Separate pages show error events, warning events, and informational events. Each page displays events in the same format.

You can adjust the number of messages that appear on the page in the **Preferences** settings.

Click a message to view event details.

Timestamp

When the event occurred.

Message originator

Internal application server class that published the event.

Message

Identifier and short description of the event.

Message details

Use the Message Details panel of the administrative console to view detailed information about errors, warnings, and informational messages.

To view these administrative console pages, click **Troubleshooting**. Expand **Runtime Messages** and click either **Runtime Error**, **Runtime Warning**, or **Runtime Information**. Click a message to display this panel.

Each message has the following general property fields.

Message

The message ID and text.

Message type

Error, Warning, or Information.

Explanation

A description of the message.

User action

What you should do about the message.

Message originator

The name of the product class that originated the message.

Source object type

The name of the component that originated the message.

Timestamp

The date and time that the message originated.

Thread ID

The thread identifier.

Node name

The name of the node of the application server that originated the message.

Server name

The name of the application server process that originated the message.

Diagnostic Provider ID

The Diagnostic Provider ID of the component that originated the message. Click on Configuration Data, State Data, or Tests to run the corresponding diagnostic action against the originating component. A Diagnostic Provider ID will not be supplied with all messages.

Chapter 5. Administering application servers

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

About this task

After you install the product, you might have to perform one or more of the following tasks. Unless the task you want to perform is dependent on the existence of an application server, you can perform these tasks in any order.

Procedure

- Create an application server.

During the installation process the product creates a default application server, named server1. You must issue either the `createApplicationServer` or `createGenericServer` `wsadmin` command from a command line to create an additional application server or generic server.

You cannot use the administrative console that is associated with the original base server to manage any additional servers that you create. You must either use command-line tools to perform these tasks for the additional servers, set up an administrative console for each server, or configure an administrative agent to provide a single interface to all of your servers, including the original base server.

gotcha: If you create additional application servers, only use one server to modify and save configurations. There is no coordination of configuration setting between the different servers and if you modify and save configurations on multiple servers, your data might become corrupted.

- Configure the server startup process such that only server components that are initially needed are started.

When the server is configured such that only the components that are initially needed are started during the startup process, the remaining components are dynamically started as they are needed.

gotcha: If you are running other WebSphere products on top of this product, make sure that those other products support this functionality before you select this property.

- Configure transport chains to handle client requests.
- Develop custom services.
- Define processes for the application server.
- Configure the Java virtual machine.

Results

Any new application servers you create are displayed in the list of servers on the administrative console Application servers page.

What to do next

- Manage your application servers. Any newly created application servers are configured with many default settings that do not display when you run the Create New Application Server wizard. You might need to change some of these settings to better fit the needs of your environment.
- Deploy an application or component on the application server.
- View the status of the applications running on the application server.

Configuring virtual hosts

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. You can separate and control which resources are available for client requests by combining multiple host machines into a single virtual host, or by assigning host machines to different virtual hosts.

Before you begin

If your external HTTP server configuration uses the default port, 9080, you do not have to perform these steps.

About this task

Virtual hosts isolate and independently manage multiple sets of resources on the same physical 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.

For example, suppose that:

- 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.
- 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.

Because the servlet is associated with a virtual host instead of the actual DNS address, The servlet on virtual host VirtualHost1 does not share its context with the servlet that has the same name on virtual host VirtualHost2. Requests for the servlet on VirtualHost1 can continue as usual, even though VirtualHost2 is refusing to fill requests for the servlet with the same name.

If any of the following conditions exist, 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:

- 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 stand-alone application servers, 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 page in the administrative console.

Perform the following steps to create a new virtual host or change the configuration of an existing virtual host.

Procedure

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 the product. 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 page, click **New**.
 - b. On the MIME type settings 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. **Servers > Server Types > Web servers**, then select the appropriate web server.
 - b. Click **Generate plug-in**, then click **Propagate plug-in**.
8. Restart the application server.

Virtual hosts

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. You can associate a virtual host to one or more Web modules, but you can associate each web module with one and only one virtual host. 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.

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

- The default alias is `*:80`, using an external port that is not secure.

- Aliases of the form *:9080 use the internal port that is not secure.
- Aliases of the form *:9443 use the secure internal port.
- Aliases of the form *:443 use the secure external port.

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.

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. A default virtual host, named `default_host`, is automatically configured the first time you start an application server. 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.

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`.

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 than default port (9080) for the application server.
- You want to use other aliases that are not listed.

When you request a resource, the product tries to map the request to an alias of a defined virtual host. The `http://host:port/` portion of the virtual host is not case sensitive, but the URL that follows is case sensitive. The match for the URL must be alphanumerically exact. 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 you 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 stand-alone 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.

Table 10. Aliases for a virtual host. 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 whenever the product 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`.

Table 11. Virtual hosts with overlapping aliases. 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.

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 stand-alone application servers 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.
- You use a Web server that listens on a port not already specified in your virtual host aliases.

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 port value can be any integer between 0 and 65535. You can also specify an asterisk (the wildcard value) if you want the system to select a free port for you.

The default reflects the value specified at product setup. The default might be 80, 81, 9060 or a similar value.

Data type	Integer
Range	0 - 65535 or the wildcard value (*)
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

Creating, editing, and deleting WebSphere variables

You can use WebSphere variables to provide settings for any of the string data type attributes that are contained in the product configuration files.

Before you begin

Because applications cannot directly access WebSphere variables, if you define a WebSphere variable inside of an application, an error message, such as "Unknown variable," is returned. If you must reference a WebSphere variable from within an application, include the following method in the application to expand the string that uses the WebSphere variable:

```
private String expandVariable(String s) throws
javax.management.JMException {
    com.ibm.websphere.management.AdminService as =
    com.ibm.websphere.management.AdminServiceFactory.getAdminService
    ();
```

```
String server = as.getProcessName();
```

```
java.util.Set result = as.queryNames(new javax.management.ObjectName("*:*,type=AdminOperations,process="
+ server), null);
```

```
return (String)as.invoke((javax.management.ObjectName)
result.iterator().next(),"expandVariable",new Object[]
{"${"+s+"}"), new String[] {"java.lang.String"});
```

Similarly, you can include the following lines of code in a script file if you want to use a script command to expand WebSphere variables.

- Using Jacl:

```
set mbean [$AdminControl completeObjectName WebSphere:*,type=AdminOperations]
$AdminControl invoke $mbean expandVariable {"${APP_INSTALL_ROOT}"}
```

- Using Jython:

```
AdminOperations = AdminControl.completeObjectName('WebSphere:*,type=AdminOperations')
print AdminControl.invoke(AdminOperations, 'expandVariable', '${APP_INSTALL_ROOT}')
```

About this task

WebSphere variables are usually used to specify file paths. The "Variable settings" topic supplies further details about specifying variables and highlights further details about product components that use them.

WebSphere variables are also used to configure:

- Product path names, such as JAVA_HOME, and APP_INSTALL_ROOT.
- Certain customization values.

The variable scoping mechanism for WebSphere variables enables you to define a variable at the node level, as well as at the server level. This mechanism enables you to specify a setting for all of the servers in a node, cluster, or cell, instead of individually specifying the setting for each server.

To define a new variable, change the value of an existing variable, or delete an existing variable complete the following steps, as appropriate.

Procedure

1. Click **Environment > WebSphere variables** in the administrative console
2. Create a new variable.
 - a. Click **New**.
 - b. Specify a name, a value, and, optionally, a description for the variable.

You can create WebSphere variables that support substitution. For example, if you enter `${<variable name>}` in the **Name** field, the value of `<variable name>` becomes the name of your new WebSphere variable. For example if you enter `${JAVA_HOME}` as the name of your variable, the name of the WebSphere variable that is created is the Java home directory.
 - c. Click **OK**.
 - d. Click **Environment > WebSphere variables** in the administrative console navigation, and verify that the variable is displayed in the list of variables for the selected scope.

The administrative console does not pick up typing errors. The variable is ignored if it is referred to incorrectly.
3. Modify the setting for an existing variable.
 - a. Click on the name of the variable that you want to change.
 - b. Modify the content of the Values field.

The Values field for some of the variables that are already defined when you install the product are read-only because changing the values that are specified for those variables might cause product processing errors.
 - c. Click **OK**.
4. Delete an existing variable.
 - a. Select the variable that you want to delete.
 - b. Click **Delete**.
 - c. Click **OK**.
 - d. Verify that this variable was removed from the list of variables for the selected scope.
5. Save your configuration.
6. Stop the affected servers and start those servers again to put the variable configuration change into effect.

If the change you made affects a node, you must stop and restart all of the servers on that node. Similarly if the change you made affects a cell, you must stop and restart all of the servers in that cell.

WebSphere variables collection

Use this page to view and change the defined product variables with their values. You can also use this page to create a new variable, or delete an existing variable. These variables are name and value pairs that are used to provide the settings for the string data type configuration attributes that are contained in one of the XML formatted configuration files that reside in the product repository.

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

To display a list of all of the variables that are defined for a specific scope, select that scope.

To view additional information about a specific variable, or to change the setting for that variable, click the variable name. Some of the pre-defined variables, that is, variables that already exist when you install the product, are set at values that are required for the product to function properly. The Value fields for these variables are read-only and cannot be edited.

To define a new variable, select the appropriate scope from the list of available options and then click **New**. The selected scope indicates the level at which the variable setting is visible.

To delete an existing variable, select the appropriate variable, and then click **Delete**. Do not delete any of the pre-defined variables. Before deleting a variable that you defined, make sure that none of your applications require the configuration attribute setting that the variable provides.

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 variable is visible on the administrative console panel. The scope is specified when a new variable is defined.

A resource can be visible in the administrative console collection table at the node or server scope.

WebSphere variables settings

Use this page to define the name and value of a WebSphere variable. A WebSphere variable is a name and value pair that is used to provide the setting for one of the string data type attributes contained in one of the XML formatted configuration files that reside in the product repository.

To view this administrative console page, click **Environment > WebSphere variables > *WebSphere_variable_name***.

Name:

Specifies the symbolic name for a product variable. After the variable is defined, this symbolic name can be specified in the **Value** field of any other product configuration field that accepts a string value. Whenever the application server encounters a configuration field that contains one or more symbolic names, it replaces the symbolic names with their defined values. For example, you might define a variable name that represent a commonly used file path or URL.

WebSphere Application Server variables are used for:

- Configuring WebSphere Application Server path names, such as *JAVA_HOME*, and *APP_INSTALL_ROOT*.
- Configuring certain customization values.

For example, `WAS_SERVER_NAME` is the pre-defined symbolic name of the variable that represents the name of the default application server that is provided with the product..

Value:

Specifies the value that the symbolic name represents.

For example, `server1` is the value of a pre-defined variable `WAS_SERVER_NAME`.

Data type String

Description:

Documents the purpose of a variable.

Data type String

Introduction: Variables

Variables come in many varieties. They are used to control settings and properties relating to the server environment. The three main types of variables that you should understand are environment variables, WebSphere 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.

Environment variables can also be specified as an application server environment entry. To specify an environment variable as an environment entry, in the administrative console, click **Servers > Server Types > WebSphere application servers** `server_name`. Then, under Server Infrastructure, click **Java process management > Process definition > Environment entries**.

WebSphere variables

WebSphere variables are name and value pairs that are used to provide settings for any of the string data type attributes contained in one of the XML formatted configuration files that reside in the product repository. After a variable is defined, the value specified for the variable replaces the variable name whenever the variable name is encountered during configuration processing.

WebSphere variables can be used to configure:

- WebSphere Application Server path names, such as `JAVA_HOME`, and `APP_INSTALL_ROOT`
- A path value for the `extendedDocumentRoot` JSP or file serving attribute. This capability enables you to add an application to each node in a clustered environment without modifying the `ibm-web-ext.xmi` file for that application on each node.

Note: For IBM extension and binding files, the `.xmi` or `.xml` file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or module. An IBM extension or binding file is named `ibm-*-ext.xmi` or `ibm-*-bnd.xmi` where `*` is the type of extension or binding file such as `app`, `application`, `ejb-jar`, or `web`. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be `.xmi`.

- For an application or module that uses Java EE 5 or later, the file extension must be .xml. If .xmi files are included with the application or module, the product ignores the .xmi files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the .xmi file name extension.

The `ibm-webservices-ext.xmi`, `ibm-webservices-bnd.xmi`, `ibm-webservicesclient-bnd.xmi`, `ibm-webservicesclient-ext.xmi`, and `ibm-portlet-ext.xmi` files continue to use the .xmi file extensions.

- Certain cell-wide customization values

To create or modify a WebSphere variable, in the administrative console click **Environment > WebSphere variables**.

A variable can apply to a node or a server.

How the variable is set determines its scope. 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.

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 > Server Types > WebSphere application servers > *server_name***, and then, in the Container settings section, click **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.

WebSphere variables

WebSphere variables are name and value pairs that are used to provide settings for any of the string data type attributes that are used to configure the product. After a variable is defined, the symbolic name that is specified for that variable can be specified in the **Value** field of any other configuration field for the product that accepts a string value.

WebSphere variables can be used to configure:

- WebSphere Application Server path names, such as `JAVA_HOME`, and `APP_INSTALL_ROOT`
- A path value for the `extendedDocumentRoot` JSP or file serving attribute. This capability enables you to add an application to each node in a clustered environment without modifying the `ibm-web-ext.xmi` file for that application on each node.

Note: For IBM extension and binding files, the .xmi or .xml file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or module. An IBM extension or binding file is named `ibm-*-ext.xmi` or `ibm-*-bnd.xml` where * is the type of extension or binding file such as `app`, `application`, `ejb-jar`, or `web`. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be `.xmi`.
- For an application or module that uses Java EE 5 or later, the file extension must be `.xml`. If `.xmi` files are included with the application or module, the product ignores the `.xmi` files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the `.xmi` file name extension.

The `ibm-webservices-ext.xmi`, `ibm-webservices-bnd.xml`, `ibm-webservicesclient-bnd.xml`, `ibm-webservicesclient-ext.xmi`, and `ibm-portlet-ext.xmi` files continue to use the `.xmi` file extensions.

- Certain cell-wide customization values

When a variable is defined, it is given a scope. The scope is the range of locations within the product network where the variable is applicable.

- 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.

The value of a configuration attribute can contain references to one or more variables. The syntax for such an attribute is the name of the variable, enclosed in either a pair of curly braces { } or a pair of parenthesis (). In either case, the variable is preceded by the dollar sign.

A string configuration attribute value can consist of:

- String literals, including the null value and an empty string
- Variable references that each includes one or more levels of indirection
- Nested variable references.
- Any combination of non-null and non-empty string literals, variable references, and nested variable references.

Table 12. WebSphere variables and attributes. The following table illustrates all of the possible combinations.

Configuration attribute consists of:	Configuration attribute value	Variable name	Second variable value	Third variable value	Fourth variable value	Expanded configuration attribute value
String literal	/IBM/ WebSphere/ AppServer	N/A	N/A	N/A	N/A	/IBM/ WebSphere/ AppServer
Variable reference	\$(WAS_ INSTALL_ ROOT)	WAS_ INSTALL_ ROOT	/IBM/ WebSphere/ AppServer	N/A	N/A	/IBM/ WebSphere/ AppServer
Variable reference with a string literal	\$(USER_ INSTALL_ ROOT)/temp	USER_ INSTALL_ ROOT	N/A	N/A	/IBM/ WebSphere/ AppServer/ profiles/ AppSrv01	/IBM/ WebSphere/ AppServer/ profiles/ AppSrv01/temp
Indirect variable reference with a string literal	\$(WAS_ INSTALL_ ROOT)/lib	WAS_ INSTALL_ ROOT	\$(MY_ INSTALL_ ROOT)	MY_ INSTALL_ ROOT	N/A	N/A

Table 12. WebSphere variables and attributes (continued). The following table illustrates all of the possible combinations.

Configuration attribute consists of:	Configuration attribute value	Variable name	Second variable value	Third variable value	Fourth variable value	Expanded configuration attribute value
Nested variable references with string literal (Example 1)	\${INSTALL_TYPE}_INSTALL_ROOT)/lib	INSTALL_TYPE	USER	USER_INSTALL_ROOT	/IBM/ WebSphere/ AppServer/ profiles/ AppSrv01	/IBM/ WebSphere/ AppServer/ profiles/ AppSrv01/lib
Nested variable references with string literal (Example 2)	\${INSTALL_TYPE}_INSTALL_ROOT)/lib	INSTALL_TYPE	WAS	WAS_INSTALL_ROOT	/IBM/ WebSphere/ AppServer/ AppServer	/IBM/ WebSphere/ AppServer/ AppServer/lib

During the configuration process, whenever a variable is encountered as the value for a configuration attribute, a variable expansion is performed on that variable. A variable expansion is the process of recursively replacing variable references with variable values until only a string literal remains as the value for the configuration attribute. If the expansion process encounters a variable that is not properly defined, the expansion of that variable stops and a `VariableExpansionException` exception is issued. The product configuration process continues. However, processing errors might occur because the value for this configuration attribute is not properly established.

gotcha: The variable expansion syntax that is provided in Versions 6.0.x, and 6.1.x, of the product, includes a variant that consists of a dollar sign, and a single letter variable name without any surrounding braces or parenthesis. This syntax is not supported in Version 8.0 or higher. All WebSphere variables references must be surrounded by matching parenthesis or braces, even if it is a single letter. That syntax required escaping of dollar signs to avoid ambiguity.

Table 13. Literal dollar sign. For backward compatibility, the escaping of the literal dollar sign is still supported, and the literal dollar sign is interpreted as indicated in the following table.

Input value	Value after expansion
\$	\$
\$\$	\$
\$\$\$	\$\$
\$\$\$\$	\$\$
\$\$\$\$\$	\$\$\$

Configuring the IBM Toolbox for Java

The IBM Toolbox for Java is a library of Java classes that are optimized for accessing IBM i data and resources. You can use the IBM Toolbox for Java JDBC driver to access local or remote DB2[®] UDB for IBM i databases from server-side and client Java applications that run on any platform that supports Java.

Before you begin

Determine which version of the IBM Toolbox for Java you want to use on your system.

About this task

The IBM Toolbox for Java is available in these versions:

IBM Toolbox for Java licensed program

The licensed program is available with every IBM i release. You can install the licensed program on your IBM i 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 IBM i Information Center: <http://publib.boulder.ibm.com/eserver/ibmi.html> 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.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 IBM Toolbox for Java JDBC driver is included with both versions of the IBM Toolbox for Java. This JDBC driver supports JDBC 3.0. For more information about IBM Toolbox for Java and JTOpen, see the product website at <http://www.ibm.com/servers/eserver/iseries/toolbox/index.html>.

Procedure

1. Install the IBM Toolbox for Java licensed program.
The licensed program is available with every IBM i release. You can install the licensed program on your IBM i 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.
2. Open the administrative console.
3. Select **Environment > WebSphere variables**.
4. In the list of available scopes, select the appropriate node.
5. Locate the WebSphere variable `OS400_TOOLBOX_JDBC_DRIVER_PATH` in the list of variables that are defined for that scope.
Depending on how many variables are defined for the selected node, you might have to navigate through multiple pages of variables to find the `OS400_TOOLBOX_JDBC_DRIVER_PATH` variable. In this situation, clicking the arrow at the bottom of the page takes you to the next page of variables for the selected node.
6. Click **OS400_TOOLBOX_JDBC_DRIVER_PATH** in the name column.
7. Set the value to the full directory path to the *jt400.jar* file downloaded in step one. Do not include *jt400.jar* in this value.
For example, if the fully qualified path to the *jt400.jar* file is:
`JDBC_Drivers/Toolbox/jt400.jar`
Specify `JDBC_Drivers/Toolbox` as the value for the `OS400_TOOLBOX_JDBC_DRIVER_PATH` variable.
8. Click **Apply** and then click **Save** to save your changes.

Managing shared libraries

Shared libraries are files used by multiple applications. Each shared library consists of a symbolic name, a Java class path, and a native path for loading Java Native Interface (JNI) libraries. You can use shared libraries to reduce the number of duplicate library files on your system.

Before you begin

Your applications use the same library files. The applications already are deployed on a server or you currently are deploying the applications.

About this task

Suppose that you have four applications that use the same library file, `my_sample.jar`. Instead of having four copies of `my_sample.jar` on your system after the four applications are deployed, you can define a shared library for `my_sample.jar` and have the four deployed applications use that one `my_sample.jar` library file.

Isolated shared libraries provide another way to reduce the number of library files. Isolated shared libraries each have their own class loader, enabling a single instance of the classes to be shared across the applications. Each application can specify which isolated shared libraries that it wants to reference. Different applications can reference different versions of the isolated shared library, resulting in a set of applications sharing an isolated shared library. With isolated shared libraries, some applications can share a single copy of Library A, Version 1 while other applications share a single copy of Library A, Version 2, for a total of two instances in memory.

Using the administrative console, you can define shared libraries for the library files that multiple applications use and then associate the libraries with specific applications or modules or with an application server. Guidelines for associating shared libraries are as follows:

- Associate a shared library file with an application or module to load the classes represented by the shared library in a local class loader, which can be an application-wide or module-wide class loader.
- Associate an isolated shared library file with an application or module to load the classes represented by the shared library in a separate class loader created for that shared library.
- Associate a shared library file with a server to load the classes represented by the shared library in a server-wide class loader. This class loader is the parent of the application class loader, and the WebSphere Application Server extensions class loader is its parent. Associating a shared library file with a server associates the file with all applications on the server.
- Do not associate an isolated shared library file with a server if you want a separate class loader for a shared library. If you associate the shared library with a server, the product ignores the isolation setting and still adds files in the shared library to the application server class loader. That is, associating an isolated shared library file with a server associates the file with all applications on the server. The product does not use an isolated shared library when you associate the shared library with a server. Associate an isolated shared library with an application or module.

Instead of using the administrative console to associate a shared library with an application, you can use an installed optional package. You 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.

Procedure

- Use the administrative console to define a shared library.

1. Create a shared library.

On a single-server product, you can define a shared library at the cell, node, or server level.

Defining a library at one of the these levels does not automatically place the library into a class loader. You must associate the library with an application, module, or server before the product loads the classes represented by the shared library into a local or server-wide class loader.

2. Associate each shared library with an application, module, or server.

- Associate a shared library with an application or module that uses the shared library file.

If you enabled the **Use an isolated class loader for this shared library** setting when creating the shared library, associate the isolated shared library with an application or module to use a separate class loader for the shared library.

- 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. Create a shared library to reduce the number of duplicate library files on your system.

Before you begin

Determine the full path name or directory of each library file for which you want a shared library.

About this task

To make a library file available to multiple applications deployed on a server, create one or more shared libraries for library files that your applications need. When you create the shared libraries, you can use variables within the library file class paths.

You can create one shared library that points to multiple files or directories. This enables you to maintain a single shared library for files that your applications need.

Or you can create a shared library for each library file that your applications need. This approach is recommended only when you have few library files and few applications that use the files. After you create a shared library, you associate it with each application that uses the library files. If you have multiple shared libraries and multiple applications that use the library files, you must complete many steps to create and associate those shared libraries. It is simpler to use one shared library for related files.

Use the Shared libraries page to create and configure shared libraries.

Procedure

1. Go to the Shared libraries page.

Click **Environment** > **Shared libraries** in the console navigation tree.
2. Select a shared library scope.

Change the scope of the collection table to see what shared libraries are in a particular cell, node or server.

 - a. Select a cell, node, or server.
 - b. Click **Apply**.

After creating a shared library, you can see whether a shared library can be used on a specific node. Select a scope to see what shared libraries are available to applications installed on or mapped to that scope.

3. Click **New**.
4. Configure the shared library.
 - a. On the shared library settings page, 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” on page 99.

To have only one instance of a version of a class shared among applications or modules, make the shared library an isolated shared library. Select **Use an isolated class loader for this shared library**. Using an isolated shared library can reduce the memory footprint when a large number of applications share the library.

- b. Click **Apply**.

What to do next

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.

If you enabled the **Use an isolated class loader for this shared library** setting when creating your shared library, associate the shared library with applications or web modules. If you associate the shared library with a server, the product ignores this setting and still adds files in the shared library to the application server class loader. The product does not use an isolated shared library when you associate the shared library with a 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.

Before you begin

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, unlike the native JVM class loader, only load native shared libraries that use the default operating system extension for the current platform. For example, on AIX, native shared libraries must end in .a when loaded by application server class loaders. The JVM class loader loads files ending in .a or .so.
- 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.

About this task

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.

Procedure

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.

If you want to associate your shared library with an application or module, also select **Use an isolated class loader for this shared library**. If you do not enable this setting, associate the shared library with an application server.

3. Associate the shared library.

- If you did not enable **Use an isolated class loader for this shared library**, associate the shared library with 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.

- If you enabled **Use an isolated class loader for this shared library**, associate the shared library with an application or module.

Associating an isolated shared library file with an application or module loads the classes represented by the shared library in a separate class loader created for that shared library. Do not associate an isolated shared library file with a server if you want a separate class loader for a shared library. If you associate the shared library with a server, the product ignores the isolation setting and still adds files in the shared library to the application server class loader. That is, associating an isolated shared library file with a server associates the file with all applications on the server.

The class loader created for an isolated shared library does not reload and, like a server class loader, exists for the lifetime of a server. For shared native libraries, you can use an isolated shared library to avoid errors resulting from reloading of native libraries.

What to do next

To verify that an application can use a shared library, test the application or examine the class loader in the Class loader viewer. Click **Troubleshooting > Class loader viewer > *module_name* > Table View**. The classpath of the application module class loader lists the classes used by the shared library.

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**.

Change the scope to see what shared libraries are in a particular node or server. By default, a shared library is accessible to applications deployed (or installed) on the same node as the shared library file. To change the scope, select the cell, a node, or a server under **Scope**.

Select a scope before you click **New** and create a shared library. After you create a shared library and map an application to the selected scope, you can associate the shared library with the application or its modules.

- To associate a shared library with an application or module, use the Shared library references page for the application. Click **Applications > Application Types > WebSphere enterprise applications > *application_name* > Shared library references**.
- To associate a shared library with a server class loader, use the settings page for the library reference for the server class loader. Click **Servers > Server Types > WebSphere application servers > *server_name* > Java and Process Management > Class loader > *class_loader_ID* > Shared library references > *shared_library_name***.

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 the level of the location of the shared library configuration file.

On single-server installations, the shared library has its configuration file in a location that pertains to the cell, node, or server level.

Data type String

Name:

Specifies a name for the shared library.

Data type String

Description:

Describes the shared library.

Data type String

Classpath:

Specifies a list of paths that the product searches for classes and resources of the shared library.

If a path in the list is a file, the product searches the contents of that Java archive (JAR) or compressed .zip file. If a path in the list is a directory, then the product searches the contents of JAR and compressed

files in that directory. For performance reasons, the product searches the directory itself only if the directory contains subdirectories or files other than JAR or compressed files.

Press Enter to separate class path entries. Entries must not contain path separator characters such as a semicolon (;) or colon (:). Class paths can contain variable names that can be substituted using a variable map.

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 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 use an isolated shared library or to be loaded within shared libraries associated with the class loader of an application server. See the **Use an isolated class loader for this shared library** setting.

Data type	String
Units	Class path

Use an isolated class loader for this shared library:

Specifies whether the shared library has a single isolated shared library shared across its associated applications or web modules.

An isolated shared library enables one instance of the library classes to be shared only among associated applications and web modules. An isolated shared library enables multiple applications or web modules to share a common set of classes across a subset of the applications. Further, an isolated shared library supports versioning and loads the minimum number of library copies. The class loader created for an isolated shared library does not reload and, like a server class loader, exists for the lifetime of a server. For shared native libraries, you can use an isolated shared library to avoid errors resulting from reloading of native libraries.

The default, `false`, is not to isolate the shared library so that each application loads its own instances of the shared library classes.

Using an isolated shared library can reduce the memory footprint when a large number of applications share the library. If you select this option, associate the shared library with applications or Web modules.

Restriction: If you associate the shared library with a server, the product ignores this setting and still adds files in the shared library to the application server class loader. The product does not

use an isolated shared library when you associate the shared library with a server. To use an isolated shared library, you must associate the shared library with applications or web modules.

Selecting this option affects the class loader order of the associated application or web module. If the class loader order for a class loader associated with an isolated shared library is **Classes loaded with the parent class loader first** (Parent first), the class loader checks whether a class can be loaded in the following order:

1. Checks whether the associated library class loaders can load the class.
2. Checks whether its parent class loader can load the class.
3. Checks whether it (application or WAR module class loader) can load the class.

If the order is **Classes loaded with the local class loader first (Parent last)**, the class loader checks in the following order:

1. Checks whether it (application or WAR module class loader) can load the class.
2. Checks whether the associated library class loaders can load the class.
3. Checks whether its parent class loader can load the class.

This setting maps to the `isolatedClassLoader` Boolean attribute of the Library object.

Boolean `false`

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.

Before you begin

This topic assumes that you have created a shared library. The shared library represents a library file used by multiple deployed applications.

You can define a shared library at the cell, node, server, or cluster level.

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.

About this task

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.

Procedure

1. If you have not done so already, map your application to a target server that is within the scope of the shared library.
2. Click **Applications > Application Types > WebSphere enterprise applications > *application_name* > Shared library references** in the console navigation tree to access the Shared library references page.
3. On the Shared library references page, select an application or module to which you want to associate a shared library.
4. Click **Reference shared libraries**.

5. On the Shared library mapping page, select one or more shared libraries that the application or modules use in the **Available** list, click >> to add them to the **Selected** list, and click **OK**.
6. Repeat steps 2 through 4 until you define a library reference instance for each shared library that your application or module requires.
7. On the Shared library references page, click **OK**.
8. Save the changes to the configuration.

Results

When you run the application, classes represented by the shared library are loaded in the application class loader.

The classes are now available to the application or module.

What to do next

To verify an association between an application and a shared library, examine the application class loader in the Class loader viewer. Click **Troubleshooting > Class loader viewer > *module_name* > Table View**. The classpath of the application module class loader lists the classes used by the shared library.

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 > Application Types > WebSphere 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 Map shared libraries and Map shared libraries to an entire application or module 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.

Before you begin

This topic assumes that you have created a shared library. The shared library represents a library file used by multiple deployed applications.

About this task

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.

Procedure

1. Configure class loaders for applications deployed on the server.
 - a. Click **Servers > Server Types > WebSphere application servers > *server_name*** to access the application server setting page.
 - b. Set values for the application **Class loader policy** and **Class loading mode** of the server.
2. Create a library reference for each shared library file that your application needs.
 - a. In the administrative console, click **Servers > Server Types > WebSphere 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 library reference settings page, 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.

What to do next

To verify that an application can use a shared library, test the application or examine the class loader in the Class loader viewer. Click **Troubleshooting** > **Class loader viewer** > *module_name* > **Table View**. The classpath of the application module class loader lists the classes used by the shared library.

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, 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 Platform, Enterprise Edition (Java EE) application is installed on a server, dependency information is specified in its manifest file. The product reads the dependency information of the application (.ear file) to automatically associate the application with an installed optional package .jar file. The product 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 the product are described 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.

The product supports using the manifest file (manifest.mf) in shared library .jar files and application .ear files. The product 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. The product 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 Java EE application or a module within a Java EE 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.

Before you begin

Read about installed optional packages in “Installed optional packages” on page 106 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.

About this task

Installed optional packages expand the existing shared library capabilities of an application server. Prior to Version 6.0, 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.

Procedure

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 shared library.
3. Assemble the application, declaring in the application manifest file dependencies to the library files named the manifest created for step 1.
4. Install the application on the server.

Results

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 > Server Types > WebSphere 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 deployed applications can use. 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 > Server Types > WebSphere 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 deployed applications can use. 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

Managing application servers

You can use either the administrative console or command-line tools to manage your application servers.

Before you begin

If you plan to change the system clock, stop the application server first. After you stop the server, change the system clock, and then restart the server. If you change the system clock on one system, you must ensure the clocks on all systems that communicate with each other, and have the product installed, are synchronized. Otherwise, you might experience errors, such as security tokens no longer being valid.

If an application server is running on an operating system when the time zone setting for the operating system is updated, the application server updates its internal time stamp. Because of a delay between the change for the time zone and the change to the application server internal time stamp, an incorrect time stamp could be posted for a file if the file is touched during this delay. The delay could be several

seconds. If the file is part of an application, this incorrect time stamp would cause the application to stop and then restart because the application server thinks that the application has been updated.

About this task

During the installation process, the product creates a default application server, named `server1`. If you create any additional application servers, you cannot start, stop, or manage these servers using the administrative console that is associated with the original base server. You must either use command-line tools to perform these tasks for the additional servers, set up an administrative console for each server, or configure an administrative agent to provide a single interface to all of your servers, including the original base server.

gotcha: If you create additional application servers, only use one server to modify and save configurations. There is no coordination of configuration setting between the different servers and if you modify and save configurations on multiple servers, your data might become corrupted.

You can perform the following steps to view and manage the default application server from the administrative console.

Procedure

1. In the administrative console click **Servers > Server Types > WebSphere 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 monitor the default server.
2. Click **server1** to view or change the configuration settings for the default server.
For example, if you do not need to have all of the sever components start during the server startup process, you might want to select **Start components as needed**, which is not selected when a new server is created. When this property is selected, server components are dynamically started as they are needed. Therefore, selecting this option can improve server startup time, and reduce the memory footprint of the server.

gotcha: Before selecting this option, verify that any other WebSphere products, that you are running in conjunction with this product, support this functionality.
3. Click **Review**, select **Synchronize changes with Nodes**.
4. Click **Save** to save any configuration changes that you made.

Results

When you click **Servers > Server Types > WebSphere application servers**, you can view the state of each server.

When you click **Servers > Server Types > WebSphere application servers > server_name**, you can view any configuration changes you made.

What to do next

You can deploy applications or components to your application servers.

Server collection

Use this topic to learn how to navigate within the administrative console to the pages where you can view information about the application servers, generic servers, Java message service (JMS) servers, and web servers that are defined for your system.

You can use these respective administrative console pages to perform the following tasks for the listed servers:

- Select one or more of the listed servers, and then click **Start** to start those servers.
- Select one or more of the listed servers, and then click one of the following options to stop those servers:

STOP When you click this option, the normal server quiesce process is followed. This process allows in-flight requests to complete before the entire server process shuts down.

Immediate Stop

This option is only available for application servers.

When you click this button, the selected sever stops but the normal server quiesce process is not followed. This shutdown mode is faster than the normal server stop processing, but some application clients might receive exceptions if an in-flight request does not complete before the server process shuts down.

Terminate

You should only click **Terminate** if the server does not respond when you click **Stop**, or, **Immediate Stop** or when you issue the Stop or ImmediateStop commands. Some application clients can receive exceptions. Therefore, you should always attempt an immediate stop before clicking **Terminate**.

- Click **New** to create a new server.
- Click **Templates** to create a new server template.
- Select one or more of the listed servers, and then click **Delete** those servers.

To view the Application servers page, in the administrative console page, click **Servers > Server Types > WebSphere application servers**. This page lists all of the application servers in the cell.

To view the web servers page, in the administrative console, click **Servers > Server Types > Web servers**. This page lists all of the web servers in your administrative domain. In addition to the previously mentioned actions, you can use this page to generate and propagate a web server plug-in configuration file.

Name

Specifies a logical name for the server. For WebSphere Application Server, server names must be unique within a node.

Node

Specifies the node on which the server resides.

Host Name

Specifies the IP address, the full domain name system (DNS) host name with a domain name suffix, or the short DNS host name for the server.

Version

Specifies the version of the product on which the server runs.

Status

Specifies whether the server is started, stopped, partially 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.

Table 14. Server status and meaning. The following table describes the server status.





	Started	The server is running.
	Partially stopped	The server is in the process of changing from a started state to a stopped state.
	Stopped	The server is not running.

Table 14. Server status and meaning (continued). The following table describes the server status.

	Unavailable	The server status cannot be determined.
---	--------------------	---

Application server settings

Use this page to configure an application server or a cluster member template. An application server is a server that provides services required to run enterprise applications. A cluster member template is the set of application server configuration settings that are assigned to new members of a cluster.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

To view this administrative console page, click **Servers > Server Types > WebSphere application servers > *server_name***.

On the **Configuration** tab, you can change field settings. You can also click **Installed applications** to view the status of applications that are running on this server. On the **Runtime** tab, you can view 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 might have different servers with the same server name as long as the server and node pair are unique. You cannot change the value that appears 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. However, you cannot have two servers named *server1* in the same node. The product uses the server name for administrative actions, such as referencing the server in scripting.

Default server1

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 when the server starts, the server components, services, and applications start on multiple threads. Setting this option to `false` specifies that when the server starts, the server components, services, and applications start on a single thread, which might lengthen start-up time.

The order in which the applications start depends on the weights that you assign to them. Applications that have the same weight start in parallel.

To set the weight of an application, in the administrative console, click **Applications > Application Types > WebSphere enterprise applications > *application_name* > Startup behavior**, and then specify an appropriate value in the **Startup order** field. The more important an application is, the lower the startup order value should be. For example, you might specify a startup order value of 1 for your most important

application, and a value of 2 for the next most important application. You might then specify a startup order of 3 for the next four applications because you want all four of those applications to start in parallel.

Data type	Integer
Default	1
Range	0 - 2147483647

Start components as needed

Select this property if you want the server components started as they are needed by an application that is running on this server.

When this property is selected, server components are dynamically started as they are needed. When this property is not selected, all of the server components are started during the server startup process. Therefore, selecting this option can improve startup time, and reduce the memory footprint of the server, because fewer components are started during the startup process.

Starting components as they are needed is most effective if all of the applications, that are deployed on the server, are of the same type. For example, using this option works better if all of your applications are web applications that use servlets, and JavaServer Pages (JSP). This option works less effectively if your applications use servlets, JSPs and Enterprise JavaBeans (EJB).

gotcha: To ensure compatibility with other WebSphere products, the default setting for this option is deselected. Before selecting this option, verify that any other WebSphere products, that you are running in conjunction with this product, support this functionality.

Access to internal server classes

Specifies whether the server can run in Restrict or Allow mode.

The Restrict mode is a diagnostic mode that you can use to help determine the suitability of applications for migration. This mode determines whether internal application server classes are accessed. The use of these internal classes might preclude the successful operation of these applications in future releases. However, the Restrict mode is not intended to exclude all classes from general use even if the classes might change. Some classes that might change are unrestricted in order to enable correct operation of the application server. The Restrict mode is not intended to provide complete isolation between an application and application server internal classes. Do not use the Restrict mode in a production runtime environment; use the results for guidance only.

The default value for this property is Allow.

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 searches in the parent class loader or in the application class loader first to load a class. The standard for Developer Kit class loaders and the product class loaders is Parent first.

This field only applies if you set the Class loader policy field to Single.

If you select Application first, 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

The process ID for this server on the native operating system.

This property is read only. The system automatically generates the value.

Cell name

The name of the cell in which this server is running.

This property is read only.

Node name

The name of the node in which this server is running.

This property is read only.

State

The runtime start state for this server.

This property is read only.

Product information

This link under Additional properties, displays the product information for your installation of the product. This information includes the product name, ID, version, build date, and build level.

From the Product Information page, you can click on the following links for additional product information:

- Components, for a list of all of the components that are installed.
- e-Fixes, for a list of all of the service updates that are installed.
- Extensions, for a list of the extensions that are installed.
- History report, for a detailed report of all installation events that have occurred since the product was installed, such as the installation of a specific service level.
- Product report, for a detailed report of the versions of the product that are installed.
- PTFs, for a list of all of PTFs that are installed.

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 > Server Types > WebSphere 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.

You can view this administrative console page, by clicking **Servers > Server Types > WebSphere application 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

Table 15. Data type. The following table describes the data type for the Port Name setting.

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.

gotcha: If your TCP/IP network is set up to use distributed dynamic virtual IP addresses (DVIPAs), and if the node agent is in the process of starting the application server, TCP/IP waits until the JVM TCP/IP timeout period expires before notifying the node agent that the target application server is not responsive.

Table 16. Data type and Default. The following table describes the data type and default for the Host setting.

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. Specifying the wildcard value is equivalent to specifying the loopback address or 127.0.0.1.

gotcha: 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, 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 channel must always be last because it accepts everything that is handed to it.

Table 17. Data type and Default. The following table describes the data type and default for the Port setting.

Data type	Integer
Default	None

Important: The following table lists server endpoints and their respective port ranges. In contrast to the z/OS® environment, for a distributed platform or the IBM i environment, the ORB_LISTENER_ADDRESS and the BOOTSTRAP_ADDRESS endpoints must not specify the same port.

Table 18. Server endpoints and their respective port ranges. The following table lists server endpoints and their respective port ranges.

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
DATAPOWERMGR_INBOUND_SECURE	1 - 65536
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.)
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

Table 18. Server endpoints and their respective port ranges (continued). The following table lists server endpoints and their respective port ranges.

Endpoint (port)	Acceptable values for the port field
WC_defaulthost	1 - 65536
WC_defaulthost_secure	1 - 65536
ORB_SSL_LISTENER_ADDRESS	Not supported for the distributed and iSeries environments

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 one of the **Custom properties** links.

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 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.

defeat: Setting custom properties at the server level is deprecated. However, you can specify a custom property for a server or the deployment manager as a WebSphere variable. Server scoped WebSphere variables still override any settings specified at the node scope, or higher, and are added to the `was.env` file.

To set a custom properties for either the deployment manager, or an application server, as an environment variable, in the administrative console, click **Environment > WebSphere variables**. You can then select the deployment manager or appropriate server from the pull-down list of available servers, nodes and cells and click **New** to create a new custom property, click on the name of an existing property to change the settings of that custom property, 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 is used.

Do not start your property names with `was.` because this prefix is reserved for properties that are predefined in the product.

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 the types of server components that a specific application server uses during application processing. The list of server components varies according to the type of applications a specific application server processes.

For example, SIP Container might be listed as a server component for an application server that handles Session Initiation Protocol (SIP) requests, while EJB Container might be listed as a server component for an application server that handles Enterprise JavaBeans (EJB) requests. However, Messaging Server might be listed as a server component for both application servers.

You can also use this page to manage the settings for these server component, as they relate to request processing. In particular, you can specify either started or stopped as the initial state for the server component when the server process starts.

To view this administrative console page, click **System administration > Deployment Manager***server_name*. Then, in the Server Infrastructure section, click **Administration > Server components**.

To view this administrative console page for a node agent, click **System administration > Node agents > node_agent_name**. Then, in the Server Infrastructure section, click **Administration > Server components**.

Type:

Specifies the server component type, such as Name Server or Messaging Server.

Server component settings:

Use this page to view or configure a server component instance.

To view this administrative console, click **Servers > Server Types > WebSphere application servers > server_name**. Then, in the Server Infrastructure section, 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

Thread pool collection

Use this page to view and manage the thread pools that an application server uses. A thread pool enables components of the server to reuse threads, which eliminates the need to create new threads at run time. Creating new threads expends time and resources.

To view this administrative console page, you can choose more than one navigational route. For example, click **Servers > Server Types > WebSphere application servers > *server_name* > Thread pools**.

To view the settings for a specific thread pool, click the name of that thread pool.

To create a thread pool, click **New** and enter the information on the resulting panel.

To delete a thread pool, select the thread pool you want to delete, then click **Delete**.

Thread pool settings:

Use this page to configure a thread pool that an application server uses. A thread pool enables components of the server to reuse threads, which eliminates the need to create new threads at run time. Creating new threads expends time and resources.

To view this administrative console page, click **Servers > Server Types > WebSphere application servers > *server_name* > Thread pools**, and select the thread pool that you need to configure.

To configure the thread pool for the ORB Service, click **Servers > Server Types > WebSphere application servers > *server_name* > Container services > ORB service**. Under Thread Pool Settings, select one of the following actions:

- Select Use the ORB.thread.pool settings associated with the Thread Pool Manager, and then click **ORB thread pool settings**, or
- Select Use the thread pool settings directly associated with the ORB service, and then click **Thread pool settings**.

Note: Because these administrative console panels display information dynamically, you might not see all of the fields listed on any particular panel.

Name:

The name of the thread pool to create. The name must be unique within the server.

This field is not displayed if you click **thread pool settings**.

Data type String

Description:

A text description of the thread pool.

This field is not displayed if you click **thread pool settings**.

Data type	String
------------------	--------

Minimum size:

Specifies the minimum number of threads to allow in the pool. When an application server starts, no threads are initially assigned to the thread pool. Threads are added to the thread pool as the workload assigned to the application server requires them, until the number of threads in the pool equals the number specified in the Minimum size field. After this point in time, additional threads are added and removed as the workload changes. However, the number of threads in the pool never decreases below the number specified in the Minimum size field, even if some of the threads are idle.

This field is not displayed if you click **thread pool settings**.

Data type	Integer
Default	50

Maximum size:

Specifies the maximum number of threads to maintain in the default thread pool.

If your Tivoli® Performance Viewer shows the Percent Maxed metric to remain consistently in the double digits, consider increasing the Maximum size. The Percent Maxed metric indicates the amount of time that the configured threads are used.

Data type	Integer
Default	50

Thread inactivity timeout:

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

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

Data type	Integer
Units	Milliseconds
Default	60000

Note: Default value was changed from 35000 milliseconds as documented in version 6.0 to the correct default of 60000 milliseconds.

Allow thread allocation beyond maximum thread size:

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

The maximum number of threads that can be created is constrained only within the limits of the Java virtual machine and the operating system. When a thread pool that is allowed to grow expands beyond the maximum size, the additional threads are not reused and are discarded from the pool after required work items are completed.

Data type	Boolean
Default	Cleared

Environment entries collection

Use this page to view and manage 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.

To view this page, in the administrative console click **Servers > Server Types > WebSphere application 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 **Servers > Server Types > WebSphere application 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

When you start an application server, a new server process starts. This new server process is based on the process definition settings of the current server configuration.

Before you begin

Before you start an application server, verify that all of the application required resources are available. You must also start all prerequisite subsystems.

If you want server components to dynamically start as they are needed by the installed applications, verify that the **Start components as needed** option is selected in the configuration settings for the application server before you start the application server. Selecting this option can improve startup time, and reduce the memory footprint of the application server. Starting components as they are needed is most effective if all of the applications that are deployed on the server are of the same type. For example, using this option works better if all of your applications are web applications that use servlets, and JavaServer Pages (JSP). This option works less effectively if your applications use servlets, JSPs, and Enterprise JavaBeans (EJB).

gotcha: To ensure compatibility with other WebSphere products, the default setting for this option is cleared. Before selecting this option, verify that any other WebSphere products, that you are running with this product, support this function.

About this task

This procedure for starting a server also typically applies to restarting a server. The one exception might be if a server fails and you want the recovery functions to complete their processing before new work being started on that server. In this situation, you must restart the server in recovery mode.

If you create any additional application servers, you cannot start, stop, or manage these servers using the administrative console that is associated with the original base server. You must either use command-line tools to perform these tasks for the additional servers, set up an administrative console for each server, or configure an administrative agent to provide a single interface to all of your servers, including the original base server. An administrative agent makes it easier to more fully administer these unfederated application servers.

gotcha: If you create additional application servers, only use one server to modify and save configurations. There is no coordination of configuration setting between the different servers and if you modify and save configurations on multiple servers, your data might become corrupted.

There are several options available for starting an application server.

Procedure

- You can use the startServer Qshell command.
- You can use the Submit Job (SBMJOB) CL command. You can run the following CL command from an IBM i command line.

```
SBMJOB CMD(CALL PGM(product_library/QWASSTRSVR) PARM('-profilePath'  
'profile_root' '-server' 'server_name')) JOB(server_name)  
JOBQ(QWAS8/QWASJOBQ) JOBQ(QWAS8/QWASJOBQ) USER(QEJBSVR) LANGID(*USRPRF)  
CNTRYID(*USRPRF) CCSID(*USRPRF) OUTQ(QWAS8/QWASOUTQ) ALWMLTTHD(*YES)
```

server_name is the name of the server that you are starting.

Results

The specified server starts. To verify that the server is in start state, in the administrative console, click **Servers > Server Types > WebSphere application servers**.

What to do next

After the server starts, deploy the applications that you want to run on this server.

If you must start an application server with standard Java debugging enabled:

1. In the administrative console, click **Servers > Server Types > WebSphere application servers**.
2. Click the name of the application server with the processes that you want to trace and debug.
3. On the Java virtual machine page, select the **Debug mode** option to start the standard Java debugger. Set **Debug mode** arguments, if they are needed.
4. Click **OK**.
5. Save the changes to a configuration file
6. Stop the application server.
7. Start the application server again as previously described.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

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 products or components require multiple locations.

app_client_root

The default installation root directory for the Application Client for IBM WebSphere Application Server is the /QIBM/ProdData/WebSphere/AppClient/V8/client directory.

app_client_user_data_root

The default Application Client for IBM WebSphere Application Server user data root is the /QIBM/UserData/WebSphere/AppClient/V8/client directory.

app_client_profile_root

The default Application Client for IBM WebSphere Application Server profile root is the /QIBM/UserData/WebSphere/AppClient/V8/client/profiles/*profile_name* directory.

app_server_root

The default installation root directory for WebSphere Application Server is the /QIBM/ProdData/WebSphere/AppServer/V8/Base directory.

java_home

Table 19. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32-bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit
64-bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit

plugins_profile_root

The default Web Server Plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/*profile_name* directory.

plugins_root

The default installation root directory for Web Server Plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V8/webserver directory.

plugins_user_data_root

The default Web Server Plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver directory.

product_library

product_lib

This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS8x (where x is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is QWAS8A. The *app_server_root*/properties/product.properties file contains the value for the product library of the installation, was.install.library, and is located under the *app_server_root* directory.

profile_root

The default directory for a profile named *profile_name* for WebSphere Application Server is the /QIBM/UserData/WebSphere/AppServer/V8/Base/profiles/*profile_name* directory.

shared_product_library

The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root

The default user data directory for WebSphere Application Server is the /QIBM/UserData/WebSphere/AppServer/V8/Base directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root

The default web server path is /www/*web_server_name*.

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.

About this task

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.

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.

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:

Procedure

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 *profile_root/bin* directory for the profile with which the server is associated.

Results

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.

About this task

To change the application server default user profile:

Procedure

1. Choose an existing user profile, or use the Create User Profile (CRTUSRPRF) command to create a 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 `-chggrp` parameter on the `enbprfwas` script. See Step 3 for more information about how to use the `enbprfwas` command

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 -dtaut 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 the authorities of that profile:


```
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. See the information about the System i[®] 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 > Server Types > WebSphere application servers** and select an application server.
- b. In the Server Infrastructure section, select **Java and process management > Process definition**.
- c. In the Additional Properties section, 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.

Results

You can now use the specified user profile to run application servers.

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.

Procedure

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 running when it should be operational, examine the status of the application on an Applications page and try restarting the application.
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.

Before you begin

Make sure you understand the impact of stopping a particular server has on your ability to handle work requests, especially if you need to maintain a highly available environment.

About this task

There are times you need to stop an application server. For example, you might have to apply service to an application running on that server, or you might want to change one of the application server's configuration settings. Use one of the following options when you need to stop an application server.

Procedure

- 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. You can initially set *delayTime* be 600 seconds and then adjust it, if necessary, to a value that is more appropriate for your environment. To more information about specifying a value for *delayTime* , see the topic Stop the WebSphere Application Server environment in the IBM i Information Center.

Results

The specified server stops as soon as requests assigned to that server finish processing. To verify that the server is in stop state, in the administrative console, click **Servers > Server Types > WebSphere application servers**.

What to do next

If you experience any problems shutting down a server, see the *Troubleshooting and support* PDF.

Changing time zone settings

In some application environments, it is important that application server components use the same time zone. You can use the administrative console or system environment variables to ensure that your application components use the correct time zone.

Before you begin

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).

Determine the scope at which you want to set the time zone value. You can set the time zone value such that is applies for an entire cell, for an entire node, or only for a specific server.

Remember that time zone IDs should include an offset and, in almost all cases, a daylight saving time zone name for consistent results. For example, specify EST5EDT for Eastern Standard Time, Daylight Savings Time.

About this task

In general cases, the time zone for application server is inherited from the time zone that is set for the operating system; Java should be inherit the time zone from the operating system, and the application server will use the time zone that is set for each Java Virtual Machine (JVM). If you need to configure a different time zone for a single JVM, you can set the TZ environment variable in the application server, modify the properties file, or specify a command-line parameter when the JVM starts.

You can change the time zone setting for your application logs for all of the processes running in a single application server, for all of the application servers running under a user profile, or for all of the JVM processes running on the WebSphere Application Server subsystem.

Procedure

- Set the time zone for each of your server processes.

1. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name* > Java process management > Process definition > Environment entries**.
2. Set a value for the TZ variable.
 - If the TZ variable is included in the list of defined variables, click **TZ**, and then specify a new time zone value in the **Value** field.
 - If the TZ variable is not included in the list of defined variables, click **New**, and then specify TZ in the **Name** field, and the appropriate time zone value in the **Value** field.

For example, if you specify TZ in the **Name** field, and EST5EDT in the **Value** field, Eastern United States is used as the time zone setting for all of your server processes.

3. Click **Apply**, and then click **Save** to save your changes.
 4. Stop and restart all of the affected application server that were running when you made the time zone changes.
- Update the user.timezone property with the appropriate time zone setting in the properties file.
 - To change the time zone setting for all application servers under one 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 a WebSphere Application Server subsystem, update the user.timezone property in the properties file, or set a system local variable for that subsystem.
 - Set the time zone with a command-line property for each JVM. For example, use the following parameter to set the time zone on the Java call:
`-Duser.timezone=time_zone_code`

Results

Your new time zone setting are in affect for the designated servers.

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.

Before you begin

Verify that extended National Language Support (NLS) is installed on your IBM i server. If extended NLS support is not already installed, install it by selecting option 21 when you install the base operating system (5769-SS1).

About this task

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.

gotcha: 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.

Procedure

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.

Results

All of the components of the application servers running under this user profile use the time zone specified for the user.timezone property.

What to do next

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 IBM i server.

Before you begin

Verify that extended National Language Support (NLS) is installed on your IBM i server. If extended NLS support is not already installed, install it by selecting option 21 when you install the base operating system (5769-SS1).

About this task

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 IBM i server or configure a locale for that server:

Procedure

1. Update the user.timezone property in the SystemDefault.properties file for your IBM i 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 IBM i server.

gotcha: 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.

Results

All of the JVM processes running on your IBM i server use the same time zone.

What to do next

Start your application servers.

Time zone IDs that can be specified for the user.timezone property

The following table lists the time zone IDs that you can specify for the user.timezone property.

- 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 IBM i 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.

Important: The United States and Canada are making changes to the Daylight Saving Time start and end dates. The Technote Changes to Daylight Saving Time will affect IBM WebSphere Application Server and its associated Operating Systems, that is available on the Support website, provides the latest information on service updates that are being made to support these changes.

Table 20. Time zone IDs for the user.timezone property. The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i 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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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
America/Atka	-10 : 00	60	Hawaii-Aleutian Standard Time	
Etc/GMT+10	-10 : 00		GMT-10:00	
HST	-10 : 00		Hawaii Standard Time	
Pacific/Fakaofu	-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/Tijuana	-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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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)	
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	

Table 20. Time zone IDs for the user.timezone property (continued). The following table lists the time zone IDs that you can specify for the user.timezone property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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	

Table 20. Time zone IDs for the `user.timezone` property (continued). The following table lists the time zone IDs that you can specify for the `user.timezone` property.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (IBM i only)
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	
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.

About this task

Run the `chgwassvr` script command from the Qshell command line to change the ports for an application server.

Procedure

1. On the IBM i command line, issue the **STRQSH** command to start the Qshell.
adequateNumberOfSeconds is the amount of time that it takes to stop the WebSphere Application Server servers that are running on your system.
2. Run the **chgwassvr** script.

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

If an application server process spontaneously closes, or web modules stop responding to new requests, it is important that you quickly determine why this stoppage is occurring. You can use some of the following techniques to determine whether the problem is a web module problem or an application server environment problem.

If an application server process spontaneously closes, or web modules running on the application server stop responding to new requests:

- Try to isolate the problem by installing the web modules on different servers, if possible.
- Use the Tivoli performance viewer to determine if any of the application server resources, such as the Java heap, or database connections, have reached their maximum capacity. If there is a resource problem, review the application code for a possible cause:
 - If database connections are being assigned to a request but are not being released when the requests finish processing, ensure that the 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.
- Enable verbose garbage collection on the application server to help you determine if you have a memory leak problems. This feature adds detailed statements about the amount of available and in-use memory to the JVM error log file.

To enable up verbose garbage collection:

1. In the administrative console, click **Servers > Server Types > Application servers > server_name**. Then, under Server Infrastructure, click **Java and process management > Process definition > Java virtual machine**, and select **Verbose garbage collection**.
2. Stop and restart the application server.
3. Periodically, browse the log file for garbage collection statements. Look for statements beginning with "allocation failure". This string indicates that a need for memory allocation has triggered a JVM garbage collection, to release unused memory. Allocation failures are normal and do not necessarily indicate a problem. However, the statements that follow the allocation failure statement show how many bytes are needed and how many are allocated. If these bytes needed statements indicate that the JVM keeps allocating more memory for its own use, or that the JVM is unable to allocate as much memory as it needs, there might be a memory leak.

You can also use the Tivoli performance viewer to detect memory leak problems.

- Browse the thread dump for clues:

The JVM creates a thread dump whenever an application server process spontaneously closes. You can also force an application to create a thread dump. After a dump is created, you can check the dump for clues as to why new requests are not being processed.

To force a thread dump:

1. Using the wsadmin command prompt, get a handle to the problem application server:

```
wsadmin>set jvm [$AdminControl completeObjectName type=JVM,process=server_name,*]
```

where *server_name* is the name of your server.

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*.

After the application creates the dump, you can check for the following clues:

- Look for an excessive current heap size. The thread dump shows information on the current Java heap size, and the minimum and maximum heap size settings.
- Look at the snapshot of each thread in the process. 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 product 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/series/allproducts/index.html>

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.

About this task

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.

Procedure

1. Configure 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 mail 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.

What to do next

Test the server and resources.

Configuring an application server to use a single network interface

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.

About this task

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.

gotcha:

- If you want a specific application server to use a single network interface, perform the following steps for that application server.
- 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.

Procedure

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 panel. Click **Servers > Server Types > WebSphere application servers > server_name > Container Settings > Container services > ORB Service**. Then in the Additional Properties section, 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. Click **Servers > Server Types > WebSphere application servers > *server_name* > Java 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. Click **Servers > Server Types > WebSphere application servers > *server_name***, and then under Communications 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 Version 5 JMS servers to Stopped .
 - a. In the administrative console, click **Servers > Server Types > Version 5 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 > Server Types > WebSphere application servers > *server_name***.
 - 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 your changes.
 - a. In the administrative console, click **System administration > Save Changes to Master Repository**.
 - b. Click **Save**.
 7. Stop and restart the application server.

Results

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.

Example

This example creates two application servers, each using a different network interface, on a machine that has at least two network interfaces:

1. Use the Profile Management tool to create an application server profile.
2. Use the Profile Management tool to create a second application server profile, specifying a host name that is different than the host name used for the previously created application server.
3. Start the application server that is configured to the first network interface. Follow the preceding steps to prepare this server to communicate on the network interface that you specified when you configured this application server.
4. Start the second application server. Follow the preceding steps to prepare this server to communicate only on the network interface you specified when you configured this second application server.
5. Stop both of the application servers that you created in this example.
6. Restart both of these application servers.

You have two separate nodes running on two different network interfaces.

What to do next

If you are using a stand-alone 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.

gotcha: Do not set *host_name* to localhost, a loop back address, such as 127.0.0.1, or an * (asterisk).

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.

About this task

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.

gotcha: 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:

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers**, and select the server that you want to enable for UCS Transformation Format.
2. Then, in the Server Infrastructure section, click **Java and process management > Process definition > Java virtual machine**.

3. Specify `-Dclient.encoding.override=UTF-8` for the **Generic JVM Arguments** property, 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.

Results

The application server uses UCS Transformation Format for encoding.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

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 products or components require multiple locations.

app_client_root

The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root

The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root

The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/profile_name` directory.

app_server_root

The default installation root directory for WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppServer/V8/Base` directory.

java_home

Table 21. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32-bit IBM Technology for Java	<code>/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit</code>
64-bit IBM Technology for Java	<code>/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit</code>

plugins_profile_root

The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/profile_name` directory.

plugins_root

The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root

The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library

product_lib

This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS8x (where x is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is QWAS8A. The *app_server_root/properties/product.properties* file contains the value for the product library of the installation, *was.install.library*, and is located under the *app_server_root* directory.

profile_root

The default directory for a profile named *profile_name* for WebSphere Application Server is the */QIBM/UserData/WebSphere/AppServer/V8/Base/profiles/profile_name* directory.

shared_product_library

The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root

The default user data directory for WebSphere Application Server is the */QIBM/UserData/WebSphere/AppServer/V8/Base* directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root

The default web server path is */www/web_server_name*.

Managing the QWAS8 subsystem for WebSphere Application Server

WebSphere Application Server runs in its own subsystem, called QWAS8, which is installed with the product. For each WebSphere Application Server profile, the number of jobs that run in the QWAS8 subsystem depends on which product and which services you are using.

About this task

You can perform the following tasks within this subsystem.

Procedure

- Prepare the subsystem to run WebSphere Application Server profiles.
- Start the WebSphere Application Server environment.

When you start the default server that is created when you install the product, SERVER1 is listed as a started job.

- Start and stop generic application servers.
- Set explicit authorities for the startServer and the stopServer scripts.

gotcha: *ALLOBJ authority must be set before you can run the startServer and the stopServer scripts from the HTTP Administration forms.

- Shut down the WebSphere Application Server subsystem.

Starting the application server environment in the QWAS8 subsystem

WebSphere Application Server runs in its own subsystem, called QWAS8, which is installed with the product. Use these steps to prepare your system to run WebSphere Application Server profiles.

About this task

WebSphere Application Server profiles run in the QWAS8 subsystem and require that TCP/IP is configured and activated on the system. To prepare your system to run WebSphere Application Server profiles:

Procedure

1. Start Transmission Control Protocol/Internet Protocol (TCP/IP). On the CL command line, enter:
STRTCP
2. Start your application server profile. If the QWAS8 subsystem is not active, it is started when you start your application server profile.

gotcha: The default application server does not automatically start when the subsystem is started. If the QWAS8 subsystem is not currently active when you use the startServer Qshell script to start your server, the QWAS8 subsystem is started.

What to do next

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(QWAS8)
```

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 System i Navigator.

Configuring application servers to automatically start when the QWAS8 subsystem starts

WebSphere Application Server runs in its own subsystem, called QWAS8, which is installed with the product. Use these steps to configure application servers to start automatically when the QWAS8 subsystem starts.

Procedure

1. Give your user profile authority to the QWAS8/QWASJOBDD job description and QWAS8/QWAS8 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(QWASJOBDD) FROMLIB(QWAS8) OBJTYPE(*JOBDD) TOLIB(mywasjobdd)
NEWOBJ(mywsserv)
 - b. Change the user of the duplicate job description (jobdd) from *RQD to QEJBSVR to avoid multiple job descriptions with the same user. For example, issue the following command on the CL command line:
CHGJOBDD JOBDD(mywasjobdd/mywsserv) USER(QEJBSVR)
 - c. Use the CHGJOBDD 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 for the application server (server1) when the subsystem is started, set the RQSDTA field as follows:

```
'QSYS/CALL PGM(product_library/QWASSTRSVR) PARM('-profilePath'  
          ''user_data_root/profiles/default'  
          ''-server'' ''server1'')
```

2. Add an autostart job entry to the QWAS8/QWAS8 subsystem. Enter the following command from the CL command line:

```
ADDAJE SBS(QWAS8/QWAS8) JOB(mywsserv) JOBDD(mywasjobdd/mywsserv)
```

3. Optional: Configure the system such that the QWAS8 subsystem starts during system startup. To enable automatic startup, add the following line to the system startup program:

```
STRSBS QWAS8/QWAS8
```

Notes:

- The system startup program is defined by the QSTRUPPGM system value.
- TCP/IP must be active before the product subsystem can start. Ensure that the STRTCP command runs before the STRSBS QWAS8/QWAS8 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 QWAS8 subsystem for WebSphere Application Server

You can shutdown the WebSphere Application Server subsystem in order to completely shutdown the WebSphere Application Server environment.

About this task

For information about stopping individual servers, nodes and deployment managers, see the topics *Starting an application server* and *Stopping an application server*. 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 QWAS8 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, and so on, 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), 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 must increase the amount of time specified on the ENDSBS or ENDJOB command.

Procedure

1. Invoke the End Subsystem (ENDSBS) CL command specifying the QWAS8 subsystem.

```
ENDSBS SBS(QWAS8) 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 QWAS8 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.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

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 products or components require multiple locations.

app_client_root

The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root

The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root

The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/profile_name` directory.

app_server_root

The default installation root directory for WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppServer/V8/Base` directory.

java_home

Table 22. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32-bit IBM Technology for Java	<code>/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit</code>
64-bit IBM Technology for Java	<code>/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit</code>

plugins_profile_root

The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/profile_name` directory.

plugins_root

The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root

The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library

product_lib

This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to `.exe`, `.dll`, `.so` objects) for the installed product. The product library name is `QWAS8x` (where `x` is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is `QWAS8A`. The `app_server_root/properties/product.properties` file contains the value for the product library of the installation, `was.install.library`, and is located under the `app_server_root` directory.

profile_root

The default directory for a profile named *profile_name* for WebSphere Application Server is the /QIBM/UserData/WebSphere/AppServer/V8/Base/profiles/*profile_name* directory.

shared_product_library

The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root

The default user data directory for WebSphere Application Server is the /QIBM/UserData/WebSphere/AppServer/V8/Base directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root

The default web server path is /www/*web_server_name*.

Creating generic servers

A generic server is a server that is managed in the WebSphere Application Server administrative domain even though the server 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 WebSphere Application Server or process.

About this task

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.

gotcha: For the Base WebSphere Application Server, you cannot use the administrative console to create a generic application server definition or use the administrative console to start, stop or, in any way, control or manage that application server. To create a generic server, use the wsadmin tool. To manage Base generic application servers, you need to use the command prompt environment, such as startServer <genericServerName> or stopServer <genericServerName> or serverStatus <genericServerName>.

Procedure

- **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 product environment. It is recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular WebSphere Application Server servers.

4. Click **Next**
5. Click **Finish**. The generic server now appears as an option on the **Generic servers** page in the administrative console.
6. On the **Generic servers** page, click on the name of the generic server.
7. Under Additional Properties, click **Process Definition**.
8. In the Executable name field, enter the name of the non-java process that is launched when you start this generic server.

For example, if you are using a perl script as a generic server, enter the path to the perl.exe module in the Executable name field.

If you have additional arguments, such as the name of the perl script and its parameters, enter them in the Executable arguments field. Multiple arguments must be separated by carriage returns. Use the Enter key on your keyboard to create these carriage returns in the Executable arguments field. The following example illustrates how a perl script application that requires two arguments should appear in this field:

```
perl_application.pl  
arg1  
arg2
```

gotcha: The 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.

9. 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 > Server Types > Generic servers**
 2. Click **New**.
 3. Type in a name for the generic server.

The name must be unique within the application server. It is highly recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular WebSphere Application Server servers.
 4. Click **Next**
 5. Click **Finish**. The generic server now appears as an option on the **Application servers** 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 value for the product 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 value for WebSphere Application Server is `com.ibm.ws.runtime.WsServer`.
 12. Click **OK**.

Note: If the generic server is to run on an application server other than a WebSphere Application Server 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.

What to do next

After you define a generic server, use the Application Server administrative console to start, stop, and monitor the associated non-WebSphere Application Server server or process when stopping or starting the applications that rely on them.

gotcha: 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.

About this task

If you create a generic server on a profile of the WebSphere Application Server (base), you cannot use the base Application Server administrative console to start or terminate this server. You must use the wsadmin tool to manage this server.

Procedure

1. Start a generic application server.
Use the launchProcess operation of the wsadmin tool to start a generic application server.
 - a. View the **Status** value and any messages or logs to see whether the generic server starts.
2. Terminate generic servers.
Use the MBean terminate launchProcess operation of the wsadmin tool to terminate a generic server.
 - a. In the administrative console, click **Servers > Server Types > Generic servers**.
 - b. Select the check box beside the name of the generic server, and then click **Terminate** or **Stop**.
 - c. View the **Status** value and any messages or logs to see whether the generic server terminates.

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 product administrative domain, although it is not a server that is provided with the 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 > Server Types > 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.

It is highly recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular product 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	

Enabling user profiles to run application servers with System i Navigator

Except for the default user profile (QEJBSVR), all user profiles that are used to run an application server must be enabled through the System i Navigator for the WebSphere Application Server.

Before you begin

The user profile that you use to run the System i Navigator must have *SECADM special authority to perform this task.

About this task

To enable a user profile to use the System i Navigator to run application servers:

Procedure

1. Open the System i Navigator.
2. Right click the icon for the WebSphere Application Server.
3. Select **Application Administration** on the 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**.

Results

The selected user profile can use the System i Navigator to run application servers.

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.

Before you begin

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.

About this task

You need to configure transport chains to provide a common networking service for all components.

You can use either the administrative console or wsadmin commands to create a transport chain. If you use the administrative console, complete the following steps:

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name*** or **Servers > Server Types > WebSphere proxy servers > *server_name***, and then select one of the following options, depending on the type of chain you are creating:

For application servers, in the Container settings section select one of the following options:

- Click **SIP Container Settings > SIP container transport chains**.
- Click **Web container settings > Web container transport chains**.
- In the Server messaging section, click either **Messaging engine inbound transports** or **WebSphere MQ link inbound transports**.

For proxy servers, under HTTP proxy server settings, click **Proxy server transports** and select either **HTTPS_PROXY_CHAIN** or **HTTP_PROXY_CHAIN**. Then click **HTTP proxy inbound channel**.

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.

gotcha: If you are configuring a chain that contains a TCP channel, the wizard displays a list of configured TCP channels and a list of the ports that the listed TCP channels are not using. You must select one of the ports that none of the other TCP channels are using.

Similarly, if you are configuring a transport chain that contains a UDP channel, the wizard displays a list of already configured UDP channels and a list of the ports that these UDP channels are not using. You must select one of the ports that none of the other UDP channels are using.

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, complete the following actions:

- 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.

What to do next

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, the product converts the call to a transport channel call.

Note: If you create a new web container transport chain, the initial value for the `writeBufferSize` attribute is 8192, which is too small for most web container transport chains. It is recommended that you use the administrative console or `wsadmin` scripting to change the value of this attribute to 32768. Complete the following steps if you want to use the administrative console, to change the value of this attribute:

1. Click **Servers** and expand **Server Types**.
2. Click **WebSphere application servers** > *server_name*.
3. Expand **Web Container Settings** and click **Web container transport settings** > *transport_chain_name*.
4. Click **Web container inbound channel**.
5. Specify 32768 in the **Write buffer size** field, and click **OK**.
6. Click **Save**.

To change the value using `wsadmin` scripting, see the documentation about working with Web container inbound channel properties files.

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.

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.

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.

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 web server plug-in, 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 service 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 web server plug-ins 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.

transition: 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 > Server Types > WebSphere 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 IBM i and distributed operating systems, there is no limit to the number of HTTP ports that are allowed per process.

SSL Enabled

Specifies whether to protect transport connections 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 is not available if you do not have an HTTP transport defined for your system.

default: 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 have HTTP transports defined for your system, in the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name***, and then in the Container Settings section, click **Web container > HTTP transports > *host_name*** to view or change the settings for your HTTP transport.

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 transport connections 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 SSL connections. The options include one or more SSL settings that are defined in the Security Center; for example, DefaultSSLSettings, ORBSSLSettings, or LDAPSSLSettings.

Data type	String
------------------	--------

HTTP transport custom properties

You can use the administrative console to set custom properties for an HTTP transport. The HTTP transport custom properties administrative console page only appears if you have an HTTP transport defined for your system.

gotcha: You can use HTTP transports only on a version previous to a Version 6.x node in a mixed cell environment. This panel shows if you are using nodes from a version previous to Version 6.x and have the script compatibility mode enabled. You must use HTTP transport channels instead of HTTP transports to handle your HTTP requests on all of your other nodes. The topic *HTTP Tunnel transport channel custom property* describes the custom properties that you can specify for an HTTP transport channel.

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 in 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 using the administrative console, complete the following steps:

1. Click **Servers > Server Types > WebSphere application servers > *server_name***.
2. Under Container Settings section, expand **Web container settings** and click **Web container > HTTP transport**.
3. Select a host.
4. In the Additional Properties section, select **Custom Properties**.
5. On the custom properties page, click **New**.
6. 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.
7. Click **Apply** or **OK**.
8. Click **Save** on the console task bar to save your configuration changes.
9. Restart the server.

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

- “ConnectionIOTimeout” on page 166
- “ConnectionKeepAliveTimeout” on page 166
- “KeepAliveEnabled” on page 166
- “MaxConnectBacklog” on page 166
- “MaxKeepAliveConnections” on page 166
- “MaxKeepAliveRequests” on page 167
- “RemoveServerHeader” on page 167
- “ResponseBufferSize” on page 167
- “ServerHeaderValue” on page 168
- “SoLingerValue” on page 168
- “TcpNoDelay” on page 168
- “Trusted” on page 168

- “UseSoLinger” on page 168

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	5 seconds for the IBM i and distributed platforms

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	5 seconds for the IBM i and distributed platforms

KeepAliveEnabled: This property is only valid for IBM i 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

MaxConnectBacklog: This property is only valid for IBM i 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 IBM i 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 IBM i 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	100 requests for the IBM i and distributed platforms

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

Attention: This custom property takes effect on the web container level only. You cannot set it on the transport level. To set this custom property, see Modifying the default web container configuration.

ResponseBufferSize:

This property is used to specify, in bytes, the default size of the initial buffer allocation for the response buffer. When the buffer fills up, a flush for this buffer space will automatically occur. If a value is not specified for this property, the default response buffer size of 32K bytes is used.

The setBufferSize() API method can be used to override the value specified for this custom property at the individual servlet level.

Data type	Integer
Default	32000 bytes

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.

Attention: This custom property takes effect on the web container level only. You cannot set it on the transport level. To set this custom property, see Modifying the default web container configuration.

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: 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

Important: This property must be set to false for Secure Sockets Layer (SSL) client certificate authentication to work.

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 set to true, 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

Transport chains collection

Use this page to view or manage transport chains. Transport chains enable communication through transport channels, 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 > Server Types > WebSphere 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 the product 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 the transport chain. If the application server is on a local machine, the host name might be localhost.

Port

Specifies the port to bind for the transport chain. The port number can be any port that is not already bound to another transport chain.

SSL Enabled

When enabled, users are notified that there is a channel that enables Secure Sockets Layer (SSL) in the listed transport 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 > Server Types** , and then click either **WebSphere application servers** or **WebSphere proxy servers**. Click a server name, and then click **Ports > View associated transports** for the port whose transport chains you want view, and then click 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 the product configuration.

Enabled

When checked, this transport chain is activated at application server or proxy server startup.

Transport channels

Lists the transport channels configured for this transport chain and their configuration settings. Click the name of a transport channel to view the configuration settings for that 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 > Server Types > WebSphere 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 within the product 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 transport 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 > Server Types > WebSphere application servers > *server_name* > Ports**. Locate the port for the HTTP channel whose settings you want to view or configure, and click **View associated transports**. Click the name of the transport chain that includes this HTTP transport, and then click the name of the HTTP transport channel.

Transport channel name

Specifies the name of the HTTP transport channel.

The name field cannot contain any of the following characters: # \ / , : ; " * ? < > | = + & % ' '

This name must be unique across all channels in your system. 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 transport chain includes multiple channels to which it might forward data. The channel in the chain that has the lowest discrimination weight is the first channel that looks at incoming data and determines whether it owns that data.

Data type Positive integer
Default 0

Read timeout

Specifies the amount of time, in seconds, that the HTTP transport channel waits for a read request to complete on a socket after the first read occurs. The read being waited for could be part of the body of the read request, such as a POST, or part of the headers, if all of the headers are not read as part of the first read that occurs on the socket for this request.

transition: The value specified for this property, in conjunction with the value specified for the Write timeout property, provides the timeout functionality that the ConnectionIOTimeout custom property provided in previous releases.

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 the response data to be transmitted. This timeout typically only occurs in situations where the writes are lagging behind new requests. This situation can occur when a client has a low data rate or the network interface card (NIC) for the server is saturated with I/O.

transition: The value specified for this property, in conjunction with the value specified for the Read timeout property, provides the timeout functionality that the ConnectionIOTimeout custom property provided in previous releases.

If some of your clients require more than 300 seconds to receive data being written to them, change the value specified for the Write timeout parameter. Some clients are slow and require more than 300 seconds to receive data that is sent to them. To ensure they are able to obtain all of their data, change the value specified for this parameter to a length of time in seconds that is sufficient for all of the data to be received. Make sure that if you change the value of this setting, that the new value still protects the server from malicious clients.

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.

transition: The value specified for this property provides the timeout functionality that the ConnectionKeepAliveTimeout custom property provided in previous releases.

Data type Integer
Default 30 seconds

Use persistent (keep-alive) connections

When selected, specifies that the HTTP transport channel connections are left open between requests. Leaving the connections open can save setup and tear down costs of sockets if your workload has clients that send multiple requests.

If your clients only send single requests over substantially long periods of time, it is probably better to disable this option and close the connections right away rather than to have the HTTP transport channel setup the timeouts to close the connection at some later time.

The default value is true, which is typically the optimal setting.

gotcha: If a value other than 0 is specified for the maximum persistent requests property, the Use persistent (keep-alive) connections property setting is ignored.

Unlimited persistent requests per connection

When selected, specifies that the number of persistent requests per connection is not limited.

Maximum persistent requests per connection

When selected, specifies that the number of persistent requests per connection is limited to the number specified for the Maximum number of persistent requests property. This property setting is ignored if the Use persistent (keep-alive) connections property is not enabled.

Change the value specified for the Maximum persistent requests parameter to increase the number of requests that can flow over a connection before it is closed. When the Use persistent connections option is enabled, the Maximum persistent requests parameter controls the number of requests that can flow over a connection before it is closed. The default value is 100. This value should be set to a value such that most, if not all, clients always have an open connection when they make multiple requests during the same session. A proper setting for this parameter helps to eliminate unnecessary setting up and tearing down of sockets.

For test scenarios in which the client will never close a socket or where sockets are always proxy or web servers in front of your application server, a value of -1 disables the processing, which limits the number of requests over a single connection. The persistent timeout still shuts down some idle sockets and protect your server from running out of open sockets.

Related Information: The behavior of persistence is the same as keep-alive connections from the HTTP Transports. The MaxKeepAliveConnections setting, which specifies the maximum number of concurrent keep alive (persistent) connections across all HTTP transports, and the thread pool size are not directly related to persistence. Persistence operates independently of the MaxKeepAliveConnections setting and thread pool size settings.

Maximum persistent requests per connection

Specifies the maximum number of persistent requests that are allowed on a single HTTP connection. You can add a value to this field only if the **Maximum persistent requests per connection** property is selected.

When the Use persistent connections option is enabled, the Maximum persistent requests parameter controls the number of requests that can flow over a connection before it is closed. The default value is 100. This value should be set to a value such that most, if not all, clients always have an open connection when they make multiple requests during the same session. A proper setting for this parameter helps to eliminate unnecessary setting up and tearing down of sockets.

For test scenarios in which the client will never close a socket or where sockets are always proxy or web servers in front of your application server, a value of -1 will disable the processing which limits the number of requests over a single connection. The persistent timeout will still shutdown some idle sockets and protect your server from running out of open sockets.

If a value of 0 or 1 is specified, only one request is allowed per connection.

Data type	Integer
Default	100

Maximum header field size

Specifies, in bytes, the maximum size for a header that can be included on an HTTP request.

Setting this property to a realistic size for your applications helps you to prevent denial of service (DoS) attacks that use large headers within an HTTP request as an attempt to make a system resource, such as the applications that handle HTTP requests, essentially unavailable to intended users.

The default for this property is 32768 bytes.

Maximum headers

Specifies the maximum number of headers that can be included in a single HTTP request.

Setting this property to a realistic number for your applications helps you to prevent denial of service (DoS) attacks that use a large number of headers within an HTTP request as an attempt to make a system resource, such as the applications that process HTTP requests, essentially unavailable to their intended users.

The default for this property is 50.

gotcha: Even if you do not change the value of this property, when you use this administrative console page to change other HTTP transport channel settings, the value specified for this property when you do your save is automatically saved to the corresponding property in the server.xml file. This change will override any value you previously set for this property in the server.xml file even if you did not intend to update the current value of this property in the server.xml file.

Limit request body buffer size

When selected, specifies that size of the body of an HTTP request is limited.

This property can be used to prevent denial of service attacks that use large HTTP requests as an attempt to make a system resource, such as the applications that process HTTP requests, essentially unavailable to their intended users.

Maximum request body buffer size

Specifies, in bytes, the maximum size limit for the body of an HTTP request. If this size is exceeded, the request is not processed.

A value can be added to this field only if the **Limit request body buffer size** property is selected.

Logging

You can use the settings in this section to configure and enable National Center for Supercomputing Applications (NCSA) access logging, or HTTP error logging. If you are running the product on z/OS, you can also use this section to configure and enable Fast Response Cache Accelerator (FRCA) logging. Enabling any of these logging services slows server performance.

If you want any of the enabled logging services to start when the server starts, click **Servers > Server Types > WebSphere application servers > server_name**. Then in the Troubleshooting section, click

HTTP error, NCSA access and FRCA logging, and select **Enable logging service at server start-up**. When this option is selected, any HTTP error, NCSA or FRCA logging service that is enabled automatically starts when the server starts.

NCSA access logging

By default, the **Use global logging service** option is selected for NCSA access logging. This setting means that the NCSA access logging settings default to the settings specified for NCSA access logging on the **HTTP error, NCSA access and FRCA logging** page in the administrative console. If you want to change these settings for this specific HTTP transport channel, expand the **NCSA Access logging** section, and select the **Use chain-specific logging** option.

After you select the **Use chain-specific logging** option, you can make the following configuration changes:

- Explicitly enable or disable NCSA access logging.
- Specify an access log file path that is different from the default path.
- Specify a maximum size for the access log file that is different from the default maximum size.
- Explicitly select the format of the NCSA access log file.

Enable access logging

When selected, a record of inbound client requests that the HTTP transport channel handles is kept in the NCSA access log file.

Access log file path

Specifies the directory path and name of the NCSA access log file. Standard variable substitutions, such as `$(SERVER_LOG_ROOT)`, can be used when specifying the directory path.

Access log maximum size

Specifies the maximum size, in megabytes, of the NCSA access log file. When this size is reached, the *logfile_name* archive log file is created. However, every time that the original log file overflows this archive file, the file is overwritten with the most current version of the original log file.

Maximum number of historical files

Specifies the maximum number of historical versions of the NCSA access log file that are kept for future reference.

NCSA access log format

Specifies in which format the client access information appears in the NCSA log file. If **Common** is selected, the log entries contain the requested resource and a few other pieces of information, but does not contain referral, user agent, and cookie information. If **Combined** is selected, referral, user agent, and cookie information is included.

Error logging

By default, the **Use global logging service** option is selected for Error logging. This setting means that the Error logging settings default to the settings that are specified for Error logging on the **HTTP error, NCSA access and FRCA logging** page in the administrative console. If you want to change these settings for this specific HTTP transport channel, expand the **Error logging** section, and select the **Use chain-specific logging** option.

After you select the **Use chain-specific logging** option, you can make the following configuration changes:

- Explicitly enable or disable HTTP Error logging.
- Specify the access log file path. This path can be different from the default path.
- Specify a maximum size for the error log file. This value can be larger or smaller than the default maximum size.
- Specify the type of error messages that you want included in the HTTP error log file.

Enable error logging

When selected, HTTP errors that occur while the HTTP channel processes client requests are recorded in the HTTP error log file.

Error log file path

Indicates the directory path and the name of the HTTP error log file. Standard variable substitutions, such as \$(SERVER_LOG_ROOT), can be used when specifying the directory path.

Error log maximum size

Indicates the maximum size, in megabytes, of the HTTP error log file. When this size is reached, the *logfile_name* archive log file is created. However, every time that the original log file overflows this archive file, this file is overwritten with the most current version of the original log file.

Maximum number of historical files

Specifies the maximum number of historical versions of the HTTP error log file that are kept for future reference.

Error log level

Specifies the type of error messages that are included in the HTTP error log file.

You can select:

Critical

Only critical failures that stop the Application Server from functioning properly are logged.

Error The errors that occur in response to clients are logged. These errors require Application Server administrator intervention if they result from server configuration settings.

Warning

Information on general errors, such as socket exceptions that occur while handling client requests, are logged. These errors do not typically require Application Server administrator intervention.

Information

The status of the various tasks that are performed while handling client requests is logged.

Debug

More verbose task status information is logged. This level of logging is not intended to replace RAS logging for debugging problems, but does provide a steady status report on the progress of individual client requests. If this level of logging is selected, you must specify a large enough log file size in the **Error log maximum size** field to contain all of the information that is logged.

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 > Server Types > WebSphere 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 IBM i 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 are available for a server to use.

Leave the Maximum open connections property set to the default value 20000, which is the maximum number of connections allowed. The transport channel service by default manages high client connection counts and requires no tuning.

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.

If client connections are being closed without data being written back to the client, change the value specified for the Inactivity timeout parameter. This parameter controls the maximum number of connections available for a server's use. Upon receiving a new connection, the TCP transport channel waits for enough data to arrive to dispatch the connection to the protocol specific channels above the TCP transport channel. If not enough data is received during the time period specified for the Inactivity timeout parameter, the TCP transport channel closes the connection.

The default value for this parameter is 60 seconds, which is adequate for most applications. You should increase the value specified for this parameter if your workload involves a lot of connections and all of these connections can not be serviced in 60 seconds.

gotcha: 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
```

gotcha: 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
```

gotcha: 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 anywhere 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
```

gotcha: 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 anywhere 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 > Server Types > WebSphere 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 the product 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 transport 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 > Server Types > WebSphere application servers > server_name**.
2. Under Container settings, click **Web container settings > Web container transport chains > isecure_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 transport 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 > Server Types > WebSphere 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 transport 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 > Server Types > WebSphere 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

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

Creating a new port

To create a new port and set up a channel chain to listen on a new port:

1. Go to the **Proxy Servers > SIP Proxy 1 > Transport Chain > UDP_SIP_PROXY CHAIN** panel and select **UDP inbound channel (UDP 1)**.
2. On the following panel, select the **Port** (i.e., PROXY SIP ADDRESS (*:5060)).
3. On the following panel, select **New**.

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 > Server Types > WebSphere 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: <pre>*.1.255.0 254.*.*.9 1.*.*.*</pre> <p>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:</p> <pre>0:*:*:0:007F:0:0001:0001 F:FF:FFF:FFFF:1:01:001:0001 1234:*:4321:*:9F9f:*:*:0000</pre>

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 > Server Types > WebSphere application servers > *server_name* > Web Container Settings > Web container > 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 transport 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	32768 bytes

HTTP transport channel custom properties

If you are using an HTTP transport channel, you can add any of the following custom properties to the configuration settings for that channel.

To add a custom property:

1. In the administrative console, click **Servers > Server Types**, and then select one of the following options, depending on the type of chain you are creating:
 - **Application servers > > *server_name***. Under Web Container Settings, click **Web container transport chains > *chain_name* > HTTP Inbound Channel > Custom Properties > New**.
 - **WebSphere Proxy servers > *server_name***. Under HTTP Proxy Server Settings, click **Proxy server transports**. Then, select either **HTTPS_PROXY_CHAIN** or **HTTP_PROXY_CHAIN**, and then click **> 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 are the descriptions of the HTTP transport channel custom properties provided with the product. These properties are not shown on the settings page for an HTTP transport channel. You can use the custom properties page to define the following properties:

- “CookiesConfigureNoCache”
- “EnableBuildBackupList” on page 184
- “HonorTransferEncoding” on page 184
- “limitFieldSize” on page 184
- “limitNumHeaders” on page 185
- “localLogFilenamePrefix” on page 185
- “RemoveServerHeader” on page 185
- “ServerHeaderValue” on page 185

CookiesConfigureNoCache

Use the CookiesConfigureNoCache property to specify whether the presence of a Set-Cookie header in an HTTP response message triggers the addition of several cache related headers. If this property is set to

true, an Expires header with a very old date, and a Cache-Control header that explicitly tells the client not to cache the Set-Cookie header are automatically added. These headers are not automatically added if this property is set to false.

This property is functionality equivalent to the `com.ibm.websphere.cookies.no.header` property that was available in previous versions of the product.

Data type	Boolean
Default	True

EnableBuildBackupList

Use the `EnableBuildBackupList` property to enable the HTTP channel to scan for the history files in the access and error logs directory, and rolling these files over with any newer log files created.

When this property is set to `true`, the HTTP Channel scans for the history files in the access and error logs directory, and rolls these files over with any newer log files created.

gotcha: After you configure the HTTP error log and the NCSA access log, make sure that the **Enable NCSA access logging** field is selected for the HTTP channels for which you want logging to occur. To verify that this field is selected for an HTTP channel, click **Servers > Application Servers > server > Web Container Transport Chains > HTTP Inbound Channel**. This setting has to be enabled before setting this custom property to `true` has any effect on the HTTP channel functionality.

Data type	Boolean
Default	False

HonorTransferEncoding

Use the `HonorTransferEncoding` property to indicate whether the HTTP transport channels should convert a chunked message to a content-length delimited message when there is only one chunk.

When this property is set to `true`, the HTTP transport channels write out the chunks instead of switching to a content-length message even if the message only consists of one chunk. There is a performance impact to this setting because the HTTP transport channels does two writes for every single-chunk message: the first write is for the message, and the second write is for the zero byte chunk that marks the end of the message

When this property is set to `false`, the HTTP transport channels convert a chunked message to a content-length delimited message when there is only one chunk. This setting improves channel performance because the channel only does one write for a single-chunk message that is converted to a content-length message.

Data type	Boolean
Default	False

limitFieldSize

Use the `limitFieldSize` property to enforce the size limits on various HTTP fields, such as request URLs, or individual header names or values. Enforcing the size limits of these fields guards against possible Denial of Service attacks. An error is returned to the remote client if a field exceeds the allowed size.

Data type	Integer
Default	32768
Range	50-32768

limitNumHeaders

Use the `limitNumHeaders` property to limit the number of HTTP headers that can be present in an incoming message. If this limit is exceeded, an error is returned to the client.

Data type	Integer
Default	500
Range	50 to 4000

localLogFilenamePrefix

Use the `localLogFilenamePrefix` 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`.

gotcha: If you specify a value for the `localLogFilenamePrefix` custom property, you must also set the `accessLogFileName` HTTP channel custom property to the fully qualified name of the log file you want to use for in process requests. You cannot specify a variable, such as `$(SERVER_LOG_ROOT)`, as the value for this custom property.

Data type	String
------------------	--------

RemoveServerHeader

Use the `RemoveServerHeader` property to force the removal of any server header from HTTP responses that the application server sends, thereby hiding the identity of the server program.

Data type	Boolean
Default	False

ServerHeaderValue

Use the `ServerHeaderValue` property to specify a header that is added to all outgoing HTTP responses if a server header does not already exist.

Data type	String
Default	WebSphere Application Server v/x.x, where x.x is the version of WebSphere Application Server that is running on your system.

HTTP Tunnel transport channel custom properties

If you are using an HTTP Tunnel transport channel, you can add the following custom properties to the configuration settings for that channel.

To add a custom property:

1. In the administrative console, click **Servers > Server Types > 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.

You can use the custom properties page to define the following HTTP tunnel transport channel custom properties:

- “pluginConfigurable”

Following is a description of the HTTP Tunnel transport channel custom property that is provided with the product. 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

TCP transport channel custom properties

If you are using a TCP transport channel, you can use TCP transport channel custom properties to configure internal TCP transport channel properties.

To add a TCP transport channel custom property, perform the following actions.

1. In the administrative console, click **Servers > Server Types**, and then follow one of the following paths:
 - **Application servers > server_name**, and then select one of the following options, depending on the type of chain you are creating:
 - Expand **SIP container settings**, and click **SIP container transport chains**.
 - Expand **Web container settings**, and click **Web container transport chains**.
 - Expand **Server messaging**, and click either **Messaging engine inbound transports** or **WebSphere MQ link inbound transports**.
 - **Proxy servers**, and then expand **HTTP proxy server settings**, and click **Proxy server transports** and select either **HTTPS_PROXY_CHAIN** or **HTTP_PROXY_CHAIN**. Then click **HTTP proxy inbound channel**
2. Select the transport chain that includes the TCP channel for which you want to specify the custom property.
3. Select the **TCP inbound channel**.
4. Click **Custom properties > New**, expand **General properties**, and 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.
5. Click **Apply** or **OK**.
6. Click **Save** to save your configuration changes.
7. Restart the server.

The following TCP transport channel custom property or properties is provided with the product. They are not shown on the settings page for a TCP transport channel.

- “listenBacklog” on page 187

listenBacklog

Use the listenBacklog property to specify the maximum number of outstanding connection requests that the operating system can 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 is rejected. The value of this property is specific to each transport.

If you need to control the number of concurrent connections, use the **Maximum open connections** field on the administrative console TCP transport channel settings page.

Data type	Integer
Default	511

Web container transport chain custom properties

Use this page to set custom properties for a web container transport channel.

To specify custom properties for a specific transport on the web container transport chain:

1. In the administrative console click **Servers > Server Types > WebSphere application servers > *server_name* > Web Container Settings > Web container transport chains**.
2. Select a transport chain.
3. Under **Transport Channels** select **Web container inbound channel (*channel_name*)**.
4. Under Additional Properties select **Custom properties**.
5. On the Custom properties page, click **New**.
6. On the settings page, enter the property that you want to configure in the Name field and the value that you want to set it to in the Value field.
7. Click **Apply** or **OK**.
8. Click **Save** on the console task bar to save your configuration changes.
9. 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. You can use the custom properties page to define the following custom properties:

- “disableRequestMessageChunking”
- “maxRequestMessageBodySize” on page 188
- “useStrictSSLConnectTimeout” on page 188

disableRequestMessageChunking

This custom property disables request message chunking when set to *true*. All of the request body up to protocol_http_large_data_inbound_buffer is buffered in memory.

For WCInboundAdmin and WCInboundAdminSecure transport chains, chunking is enabled by default to install large EAR files through the administrative console. For example, the settings for these chains are *disableRequestMessageChunking=false*. When chunking is enabled, the protocol_http_large_data_inbound_buffer value is ignored because the entire HTTP request is not buffered in the controller.

When chunking is disabled, the protocol_http_large_data_inbound_buffer value is used because the entire HTTP request is buffered in the controller.

Property name	disableRequestMessageChunking
Data type	string
Value	True or False

Defaults

By default, administrative chains have the `disableRequestMessageChunking` custom property explicitly set to `true`.

maxRequestMessageBodySize

When `disableRequestMessageChunking` is set to `false`, this is the maximum amount of request body that is buffered in memory before sending the next chunk to the servant. The `maxRequestMessageBodySize` custom property is valid only if the `disableRequestMessageChunking` custom property is set to `false`.

Property name

`maxRequestMessageBodySize`

Value

The default value is 32 kilobytes (KB). The minimum value is 32 and the maximum value is 8192, which is equivalent to 8MB.

useStrictSSLConnectTimeout

When this property is set to `true`, during a handshake with the client, the SSL Channel calculates the amount of time that can elapse before a the TCP timeout occurs based on the setting for the Socket timeout on the TCP channel. Therefore, when this property is set to `true`, the handshake can never take longer than the amount of time specified for the Socket timeout on the TCP Channel.

This property only applies to the SSL channel for a secure Web container transport chain, and is set by clicking the *name of the transport chain* > **SSL inbound channel** > **Custom properties**.

Property name

`useStrictSSLConnectTimeout`

Value

The default value is `false`.

Configuring inbound HTTP request chunking

Inbound HTTP request chunking is used to eliminate the restriction on messages greater than 10MB. The 10MB restriction is set because the entire message is buffered in the controller before the HTTP request is dispatched to the servant, therefore, the controller may fail with an out of memory condition when multiple large HTTP messages are processed simultaneously. With chunking enabled, the message is broken up into smaller pieces before it is processed by the web container and application. As a result, only one small chunk is buffered in memory at a time in the controller thus greatly reducing the amount of memory consumed by large HTTP messages. Applications do not require changes to enable inbound HTTP chunking.

About this task

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. When chunking is enabled for a particular chain, you can also configure the maximum chunk size for chunking enabled for each chain.

All HTTP web container chains have chunking enabled by default.

Procedure

1. In the administrative console click **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **Web Container Settings** > **Web container transport chains**.
2. Select a transport chain.
3. Under Transport Channels select **Web container inbound channel (channel_name)**.
4. Under Additional Properties select **Custom properties** to configure inbound HTTP request message chunking. See the article, “Web container transport chain custom properties” on page 187 for details about request message chunking settings.

- a. If the `disableRequestMessageChunking` property is already defined, select the **`disableRequestMessageChunking`** property from the list.
 - b. If the `disableRequestMessageChunking` property is not defined, click **new**.
5. On the settings page, do one of the following:
- To enable request message chunking, enter the property, **`disableRequestMessageChunking`** in the Name field and the enter the value, `false`, in the Value field. Click **Apply** or **OK** so save the custom property changes.
 - To disable request message chunking, enter the property, **`disableRequestMessageChunking`** in the Name field and the enter the value, **`true`**, in the Value field. Click **Apply** or **OK** so save the custom property changes.
6. Configure message chunk size if request message chunking is enabled. See the article, “Web container transport chain custom properties” on page 187 for details on these settings.
- a. On the Custom Properties page, click **New**.
 - b. On the settings page, enter the property, **`maxRequestMessageBodySize`**, in the Name field and the enter a size, specified in kilobytes, between 32 and 8192 in the Value field.
 - c. Click **Apply** or **OK**.
7. Click **Save** on the console task bar to save your configuration changes.
8. 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 an 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.

Before you begin

Determine whether you want to delete a particular transport chain or all of the transport chains that are associated with a specific port.

About this task

You might have to delete one or more transport chains if you have to delete a port.

To delete a transport chain:

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name* > 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.

What to do next

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.

Before you begin

Determine whether you want to disable a particular transport chain or all of the transport chains that are associated with a specific port.

About this task

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:

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name* > 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.

What to do next

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.

Creating custom services

You can create one or more custom services for an application server. Each custom services defines a class that is loaded and initialized whenever the server starts and shuts down. Each of these classes must implement the `com.ibm.websphere.runtime.CustomService` interface. After you create a custom service, use the administrative console to configure that custom service for your application servers.

About this task

Following is a list of restrictions that apply to the product custom services implementation. Most of these restrictions apply only to the initialize method:

- The initialize 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 initialize and shutdown methods are called only once on each service, and once for each operating system process that makes up the server instance.
- 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 Java Platform, Enterprise Edition (Java EE) 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 Java EE 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.
- JNDI operations that request resources are not supported.

Procedure

1. Develop a custom service class that implements the `com.ibm.websphere.runtime.CustomService` interface.

The `com.ibm.websphere.runtime.CustomService` interface includes an initialize and shutdown methods. The application server uses the initialize method to pass properties to the custom service. These properties can include:

- A property that provides the name of an external file that contains configuration information for the service. You can use the `externalConfigURLKey` property to retrieve this information.
- Properties that contain name-value pairs that are stored for the service, along with the other system administration configuration data for the service.

Both the initialize and shutdown methods declare that they might create an exception, although no specific exception subclass is defined. If either method creates an exception, the runtime logs the exception, disables the custom service, and continues to start the server.

2. Configure the custom service.

In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name***, 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. This procedure adds the path name to the extension classloader, which allows the custom service to locate and correctly load the native library.

3. Stop the application server, and then restart it.

If you are developing a custom service for an application server, stop the application server, and then restart the server.

Results

Each custom services defines a class that is loaded and initialized whenever the server starts and shuts down.

The custom service loads and initializes whenever the server starts and shuts down.

Example

As previously mentioned, your custom services class must implement the `com.ibm.websphere.runtime.CustomService` interface. In addition, your class must implement the `initialize` and `shutdown` methods. The following example, shows the code that declares the class `ServerInit` that implements your custom service. This code assumes that your custom service class needs a configuration file. This example also includes the code that accesses the external configuration file. If your class does not require a configuration file, you do not have to include the `configProperties` portion of this code.

```
public class ServerInit implements com.ibm.websphere.runtime.CustomService
{
/**
 * The initialize method is called by the application server runtime when the
 * server starts. The Properties object that the application server passes
 * to this method must contain all of the configuration information that this
 * service needs 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 runtime when the
 * server begins its shutdown processing.
 *
 * public void shutdown() throws Exception
 * {
 *     // Implement shutdown method
 * }
}
```

What to do next

Check the application server to verify that the `initialize` and `shutdown` methods of the custom service run the way that you want them to run.

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 an 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.

About this task

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.

You can define application server processes using the administrative console or the wsadmin tool.

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers**, and then click on an application server name.
2. In the Server Infrastructure section, click **Java and process management > Process definition**.
3. 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**.
4. Specify process execution statements for starting or initializing a UNIX or IBM i process.
5. Specify monitoring policies to track the performance of a process.
6. 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.
7. Specify name-value pairs for properties needed by the process definition.

gotcha: 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.

8. Optional: Prevent the application server from creating javacore dumps.

A javacore dump, or a thread dump as it is also called, is one of the primary problem determination documents that an application server creates. Also, the performance impact of creating a javacore dump is usually ignorable. Therefore, in most product environments, you should not suppress the creation of a javacore dump.

In certain circumstances, such as when there are security consideration, you might want to prevent the application server from creating javacore dumps. To disable the javacore dump function:

- a. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name***, and then in the Server Infrastructure section, click **Java and process management > Process definition > Java virtual machine > Custom properties > New**
 - b. In the Name field enter `DISABLE_JAVADUMP` and in the Value field, enter `true` to prevent the application server from creating javacore dumps.
9. Stop the application server, and then have the executable, that the process definition specifies, restart the server. If the executable cannot restart the application server, the executable should use the generic server.
 10. Check the server to verify 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.

For the WebSphere Application Server and the WebSphere Application Server, Express products, only the command-line information for starting or initializing a process applies.

To view this administrative console page, click **Servers > Server Types > WebSphere application servers > *server_name***. Then, in the Server Infrastructure section, click **Java and process management > Process definition**.

Executable name

This command line information specifies the executable name that is invoked to start the process.

For example, if you are using a perl script as a generic server, enter the path to the perl.exe module in the Executable name field, and then enter the name of the perl script, along with any arguments, in the Executable arguments field.

Table 23. Data type. The following table describes the data type for the Executable name setting.

Data type	String
-----------	--------

Executable arguments

This command line information specifies the arguments that are passed to the executable when starting the process.

You can enter multiple arguments in this field, but they must be separated by carriage returns. Use the Enter key on your keyboard to create these carriage returns. The following example illustrates how a perl script application that requires two arguments should appear in this field:

```
perl_application.pl
arg1
arg2
```

Table 24. Data type and units. The following table describes the data type and units for the Executable arguments setting.

Data type	String
Units	Command-line arguments

Start command (startCommand)

This command line information specifies the platform-specific command to launch the server process.

Start command arguments (startCommandArgs)

This command line information specifies any additional arguments required by the start command.

If you have two or more arguments that need to be passed for process definition settings, then you must specify the arguments on separate lines. For example, if you are specifying port 8089 and a configuration file (location and file name) as command arguments, you would specify:

```
<startCommandArgs>8089<startCommandArgs>
<startCommandArgs>/opt/payexpert/conf/PCPILogServer.properties<startCommandArgs>
```

In the administrative console, you specify this by entering each argument on a new line.

Note: Do not separate the command arguments with just a space.

Stop command (stopCommand)

This command line information specifies the platform-specific command to stop the server process

Table 25. Data type, Format, Example. Specify two commands in the field, one for the Stop command, and one for the Immediate Stop (CANCEL) command.

Data type	String
Format	STOP <i>server_short_name</i> ;CANCEL <i>server_short_name</i>
z/OS example	STOP BBOS001;CANCEL BBOS001

Stop command arguments (stopCommandArgs)

This command line information specifies any additional arguments required by the stop command.

Table 26. Data type, Format, Example. Specify arguments for the Stop command and the Immediate Stop (CANCEL) command.

Data type	String
Format	<i>stop command arg string</i> ;immediate stop command arg string
z/OS 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.

Table 27. Data type, Format, Example. Specify arguments for the terminate command.

Data type	String
Format	FORCE <i>server_short_name</i>
z/OS example	FORCE BBOS001

Terminate command arguments (terminateCommandArgs)

This command line information specifies any additional arguments required by the terminate command.

The default is an empty string.

Table 28. Data type, Format, Example. Specify additional arguments for the terminate command.

Data type	String
Format	<i>terminate command arg string</i>
z/OS example	ARMRESTART

Working directory

Specifies the file system directory that the process uses as its current working directory. This setting only applies for IBM i and distributed platforms. The process uses this directory to determine the locations of input and output files with relative path names.

Table 29. Data type. The following table describes the data type.

Data type	String
-----------	--------

Executable target type

Specifies 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.

Table 30. Data type. The following table describes the data type.

Data type	String
-----------	--------

Process execution settings

Use this page to view or change the process execution settings for a server process.

A server process applies to a specific application server.

If you are running on IBM i or a distributed operating systems, to view this administrative console page for an application server, click **Servers > Server Types > WebSphere application servers > server_name**. Then, in the Server Infrastructure section, click **Java and process management > 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

UMASK:

Specifies the user mask under which the process runs (the file-mode permission mask).

The deployment manager and application servers must run with a 007 umask in order to support system management functions. Therefore, it is recommended that you do not change the default value of this setting for the deployment manager or the controller.

Data type	Integer
Default	007

Run As User:

Specifies the user that the process runs as. This user ID must be defined to the security system.

Note: For the Application Server to transition to the user that is specified in this option, the user that launching the process must be a root user. This is a restriction of the operating system.

Data type

String

For the IBM i operating system, additional steps are required 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.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/index.html> and navigate to the WebSphere Application Server for iSeries Security information.

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 that is running IBM i or a distributed operating system, click **Servers > Server Types > WebSphere application servers > *server_name***, 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

Automatically restarting server processes

There are several server processes that the operating system can monitor and automatically restart when the server processes stop abnormally.

Before you begin

The Installation wizard grants you the user rights if your user ID is part of the administrator group.

About this task

You can use this function to automatically restart base servers. You can restart the **server1** process, for example.

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers > server_name > Process**. Start the administrative console. In the topology tree, expand Servers and click Application Servers. Click the name of the application server that you want to start automatically. Click Process Definition. Click Monitoring Policy. Change the Node Restart state to RUNNING. Click Apply. Save the configuration.
2. Select the application server that you want to automatically restart and then, under Server Infrastructure, click **Java and process management > Monitoring Policy**.
3. Select **Automatic restart**.
4. Click **Apply** and then click **Save** to save the change directly to the master configuration.

Results

Processes started by a **startServer** 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.

What to do next

After the process is set up, the operating system can monitor each server process and restart the process if it stops.

Return to the Defining application server processes administrative console page to continue.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

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 products or components require multiple locations.

app_client_root

The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root

The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root

The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/profile_name` directory.

app_server_root

The default installation root directory for WebSphere Application Server is the /QIBM/ProdData/WebSphere/AppServer/V8/Base directory.

java_home

Table 31. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32-bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit
64-bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit

plugins_profile_root

The default Web Server Plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/*profile_name* directory.

plugins_root

The default installation root directory for Web Server Plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V8/webserver directory.

plugins_user_data_root

The default Web Server Plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver directory.

product_library

product_lib

This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS8x (where x is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is QWAS8A. The *app_server_root*/properties/product.properties file contains the value for the product library of the installation, was.install.library, and is located under the *app_server_root* directory.

profile_root

The default directory for a profile named *profile_name* for WebSphere Application Server is the /QIBM/UserData/WebSphere/AppServer/V8/Base/profiles/*profile_name* directory.

shared_product_library

The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root

The default user data directory for WebSphere Application Server is the /QIBM/UserData/WebSphere/AppServer/V8/Base directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root

The default web server path is /www/*web_server_name*.

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).

About this task

The JVM is an interpretive computing engine that is 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 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 process, use the Java virtual machine page of the administrative console or use wsadmin scripts to change the configuration.

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name***.
2. Under Server Infrastructure, click **Java and process management > Process definition**.
3. Select **Java virtual machine**.
4. Specify values for the JVM settings as needed, and click **OK**. For more information, see the documentation about Java virtual machine settings.

Note: Java 5.0 SR10 and Java 6 SR5 correct issues in which the Java virtual machine (JVM) does not shut down correctly. If you have an application that depends on the previous behavior, which is not correct, you can revert to the previous behavior by adding the `-XXallowvmshutdown:false` argument to the Generic JVM arguments section.

5. Click **Save** in the messages section of the administrative console panel to save the changes to the master configuration.
6. Restart the application server.

Example

“Configuring application servers for UCS Transformation Format” on page 151 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 206 provides an example that involves defining a property for the JVM.

Java virtual machine settings

Use this page to view, and change the Java virtual machine (JVM) configuration 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 IBM i and distributed platforms, click **Servers > Server Types > WebSphere application servers > *server_name***. Then, in the Server Infrastructure section, click **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.

If you need to add a classpath to this field, enter each classpath entry into a separate table row. You do not have to add a colon or semicolon at the end of each entry.

The only classpaths that should be added to this field are the ones that specify the location of the following items:

- An inspection or monitoring tool to your system.
- JAR files for a product that runs on top of this product.
- JVM diagnostic patches or fixes.

Processing errors might occur if you add classpaths to this field that specify the location of the following items:

- JAR files for resource providers, such as DB2. The paths to these JAR files should be added to the relevant provider class paths.
- A user JAR file that is used by one or more of the applications that you are running on the product. The path to this type of JAR file should be specified within each application that requires that JAR file, or in server-associated shared libraries.
- An extension JAR file. If you need to add an extension JAR file to your system, you should use the `ws.ext.dirs` JVM custom property to specify the absolute path to this JAR file. You can also place the JAR file in the `WAS_HOME/lib/ext/` directory, but using the `ws.ext.dirs` JVM custom property is the recommended approach for specifying the path to an extension JAR file.

Data type

String

Boot classpath

Specifies bootstrap classes and resources for JVM code. This option is only available for JVM instructions that support bootstrap classes and resources.

If you need to add a classpath to this field, enter each classpath entry into a table row. You do not need to add the colon or semicolon at the end of each entry.

If you need to add multiple classpaths to this field, you can use either a colon (:) or semi-colon (;), depending on which operating system the JVM resides, to separate these classpaths.

The only classpaths that should be added to this field are the ones that specify the location of the following items:

- An inspection or monitoring tool to your system.
- JAR files for a product that runs on top of this product.
- JVM diagnostic patches or fixes.

Processing errors might occur if you add classpaths to this field that specify the location of the following items:

- JAR files for resource providers. such as DB2. The paths to these JAR files should be added to the relevant provider class paths.
- A user JAR file that is used by one or more of the applications that you are running on the product. The path to this type of JAR file should be specified within each application that requires that JAR file, or in server-associated shared libraries.
- An extension JAR file. If you need to add an extension JAR file to your system, you should use the `ws.ext.dirs` JVM custom property to specify the absolute path to this JAR file. You can also place the JAR file in the `WAS_HOME/lib/ext/` directory, but using the `ws.ext.dirs` JVM custom property is the recommended approach for specifying the path to an extension JAR file.

Verbose class loading

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

When this field is enabled, a report is written to the output stream each time the garbage collector runs. This report should give you an indication of how the Java garbage collection process is functioning.

You can check the verboseGC report to determine:

- How much time the JVM is spending performing garbage collection.

Ideally, you want the JVM to spend less than 5 percent of its processing time doing garbage collection. To determine the percentage of time the JVM spends in garbage collection, divide the time it took to complete the collection by the length of time since the last AF and multiply the result by 100. For example:

$83.29/3724.32 * 100 = 2.236$ percent

If you are spending more than 5 percent of your time in garbage collection and if garbage collection is occurring frequently, you might need to increase your Java heap size.

- If the allocated heap is growing with each garbage collection occurrence.

To determine if the allocated heap is growing, look at the percentage of the heap that is remains unallocated after each garbage collection cycle, and verify that the percentage is not continuing to decline. If the percentage of free space continues to decline you are experiencing a gradual growth in the heap size from garbage collection to garbage collection. This situation might indicate that your application has a memory leak.

Note: Version 7.0 and previous versions use the optthruput garbage collection algorithm. In Version 8.0, the default is set to the generational garbage collector. This garbage collection algorithm can increase performance. The following JVM option is added to the WebSphere Application Server startup command: `-Xgcpolicy:gencon`. If you prefer to use the optthruput garbage collection algorithm, you can remove `-Xgcpolicy:gencon` and the default optthruput garbage collection algorithm is used.

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, in megabytes, the initial heap size available to the JVM code. If this field is left blank, the default value is used.

For IBM i and distributed platforms, the default initial heap size is 50 MB.

bprac: These default values are sufficient for most applications.

gotcha: For IBM i, the initial heap size must always be less than the maximum heap size. Never set the initial heap size and maximum heap size properties to the same value.

Increasing this setting can improve startup. The number of garbage collection occurrences are reduced and a 10 percent gain in performance is achieved.

Increasing the size of the Java heap continues to improve throughput until the heap becomes too large to reside in physical memory. If the heap size exceeds the available physical memory, and paging occurs, there is a noticeable decrease in performance.

Maximum heap size

Specifies, in megabytes, the maximum heap size that is available to the JVM code. If this field is left blank, the default value is used.

The default maximum heap size is 256 MB. This default value applies for both 32-bit and 64-bit configurations.

Increasing the maximum heap size setting can improve startup. When you increase the maximum heap size, you reduce the number of garbage collection occurrences with a 10 percent gain in performance.

Increasing this setting usually improves throughput until the heap becomes too large to reside in physical memory. If the heap size exceeds the available physical memory, and paging occurs, there is a noticeable decrease in performance. Therefore, it is important that the value you specify for this property allows the heap to be contained within physical memory.

bprac: These default values are appropriate for most applications. Enable the **Verbose garbage collection** property if you think garbage collection is occurring too frequently. If garbage collection is occurring too frequently, increase the maximum size of the JVM heap.

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`.

Debug mode

Specifies whether to run the JVM in debug mode. The default is to not 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 the **Debug mode** property is set to `true`.

If you enable debugging on multiple application servers, verify that the same value is not specified for the address argument. The address argument defines the port that is used for debugging. If two servers, for which debugging is enabled, are configured to use the same debug port, the servers might fail to start properly. For example, both servers might still be configured with the debug argument `address=7777`, which is the default value for the debug address argument.

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.

You can enter the following optional command-line arguments in the **Generic JVM arguments** field. If you enter more than one argument, enter a space between each argument.

gotcha: If the argument states that it is only for the IBM Developer Kit only, you cannot use that argument with the JVM from another provider, such as the Microsoft or Hewlett-Packard

- **-Dcom.ibm.CORBA.RequestTimeout=*timeout_interval***

Specify the `-Dcom.ibm.CORBA.RequestTimeout=timeout_interval` argument to set 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.

Specify this argument only if your application is experiencing problems with timeouts. There are no recommended values for this argument.

- **-Dcom.ibm.server.allow.sigkill=**

The `-Dcom.ibm.server.allow.sigkill=true` argument allows the node agent process to use the terminate method of a process when the stop method does not complete within the time interval specified for the Ping interval. This setting is useful when the node agent is monitoring an application server and loses contact with that application server.

When the monitoring policy for the application server allows the node agent to restart the application server because automatic restart is enabled for the application server, the node agent executes the stop method on the application server process. During stop processing, the node agent monitors the application server and if the application server does not stop within the time interval specified for the Ping interval, and this argument is set to `true`, which is the default value, the node agent executes the terminate method on the application server process to stop the application server process.

If you set this argument to `false`, the node agent continues to monitor the stop process, but does not try to restart the application server.

To use the administrative console to disable this argument, click **System Administration > Node agents > *nodeagent_name* > Java & Process Management > Process Definition > Java Virtual Machine > Generic JVM Arguments**.

- **-Dcom.ibm.websphere.wlm.unusable.interval=*interval***

This argument only applies for z/OS. Specify the `-Dcom.ibm.websphere.wlm.unusable.interval=interval` argument to change the value of the `com.ibm.websphere.wlm.unusable.interval` property when the workload management state of the client is refreshing too soon or too late. This property specifies, in seconds, the amount of time 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. . 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.

- **-XXallowvmshutdown:false**

Use the **-XXallowvmsshutdown:false** argument to revert to a previous behavior for the JVM that is not correct. Java 5.0 SR10 and Java 6 SR5 correct issues in which the Java Virtual Machine (JVM) does not shut down correctly. If you have an application that depends on the old behavior, you can revert to the previous behavior by adding the this argument to the Generic JVM arguments section.

Data type	String
Units	Java command-line arguments

Executable JAR file name

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

Data type	String
Units	Path name

Disable JIT

Specifies whether to disable the just-in-time (JIT) compiler option of the JVM code.

If you disable the JIT compiler, throughput decreases noticeably. Therefore, for performance reasons, keep JIT enabled.

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

Operating system name

Specifies JVM settings for a given operating system.

When the process starts, the process uses the JVM settings that are specified for the server as the JVM settings for the operating system.

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*.

About this task

depfat: 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 to use the context root for that the application uses for `sendRedirect()` calls instead of using the document root for the web server, configure the Java Virtual Machine (JVM) by setting the `com.ibm.websphere.sendredirect.compatibility` property to a `true` or `false` value.

Procedure

1. Access the settings page for a property of the JVM.
 - a. In the administrative console, click **Servers > Server Types > Application servers**.
 - 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, in the Server Infrastructure section, 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 `com.ibm.websphere.sendredirect.compatibility` in the **Name** field, and either `true` or `false` in the **Value** field. Then click **OK**.
3. Click **Save** on the console task bar.
4. Stop the application server, and then restart the application server.

Java virtual machine custom properties

You can use the administrative console to change the values of Java virtual machine (JVM) custom properties.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log`, `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

To set custom properties, connect to the administrative console and navigate to the appropriate Java virtual machine custom properties page.

Application server	Click Servers > Server Types . Select either WebSphere application servers > <i>server_name</i> or WebSphere proxy servers > <i>server_name</i> , and then, under Server Infrastructure, click Java and process management > Process definition > Java virtual machine > Custom properties .
--------------------	--

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.

You can use the Custom properties page to define the following properties for use by the Java virtual machine.

- “allowDeployerRoleGenPluginCfg” on page 210
- “com.ibm.config.eclipse.wtp.enablejemtrim” on page 211
- “com.ibm.config.eclipse.wtp.enablexmltrim” on page 211
- “com.ibm.config.eclipse.wtp.jem=finer” on page 211
- “com.ibm.config.eclipse.wtp.xmltrim=finer” on page 212
- “com.ibm.eclipse.wtp.allowRootedEntries” on page 212
- “com.ibm.ejs.ras.writeSystemStreamsDirectlyToFile” on page 212
- “com.ibm.ejs.sm.server.quiesceInactiveRequestTime” on page 212
- “com.ibm.ejs.sm.server.quiesceTimeout” on page 212
- “com.ibm.websphere.deletejspclasses” on page 212
- “com.ibm.websphere.deletejspclasses.delete” on page 213
- “com.ibm.websphere.deletejspclasses.update” on page 213
- “com.ibm.websphere.ejb.UseEJB61FEPScanPolicy” on page 213
- “com.ibm.websphere.ejbcontainer.expandCMPCFJNDIName” on page 213
- “com.ibm.websphere.ejbcontainer.includeRootExceptionOnRollback” on page 213

- “com.ibm.websphere.jaxrpc.stub.typemapping.per.thread” on page 214
- “com.ibm.websphere.jaxrs.server.DisableIBMJAXRSEngine” on page 214
- “com.ibm.websphere.management.application.fullupdate” on page 214
- “com.ibm.websphere.management.application.fullupdate.*application_name*” on page 214
- “com.ibm.websphere.management.application.keepExistingSharedLibraries” on page 215
- “com.ibm.websphere.management.application.persistWebContext” on page 215
- “com.ibm.websphere.management.application.sync.recycleappasv5” on page 215
- “com.ibm.websphere.management.application.updateClusterTask.serverStopWaitTimeout” on page 215
- “com.ibm.websphere.management.application.updateSync.appExpansionTimeout” on page 216
- “com.ibm.websphere.management.configservice.validatePropNames” on page 216
- “com.ibm.websphere.management.processEmbeddedConfigGlobal” on page 216
- “com.ibm.websphere.metadata.ignoreDuplicateRefBindingsInWebModul” on page 217
- “com.ibm.websphere.network.useMultiHome” on page 217
- “com.ibm.websphere.sib.webservices.useTypeSoapArray” on page 217
- “com.ibm.websphere.webservices.attachment.tempfile.expiration” on page 218
- “com.ibm.websphere.webservices.attachments.maxMemCacheSize” on page 218
- “com.ibm.websphere.webservices.DisableIBMJAXWSEngine” on page 219
- “com.ibm.websphere.webservices.http.OneWayConnectionRecycleTime” on page 219
- “com.ibm.websphere.webservices.http.waitingThreadsThreshold” on page 219
- “com.ibm.websphere.webservices.jaxrpc.client.publishwsdl” on page 219
- “com.ibm.websphere.webservices.soap.enable.legacy.get.behavior” on page 220
- “com.ibm.websphere.webservices.tempAttachDir” on page 220
- “com.ibm.websphere.webservices.transport.jms.messageType” on page 220
- “com.ibm.websphere.webservices.transport.OPTIMIZE_HTTP_HEADERS” on page 220
- “com.ibm.websphere.webservices.transport.ssl.loadFromPolicyBinding” on page 221
- “com.ibm.websphere.webservices.UseWSFEP61ScanPolicy” on page 221
- “com.ibm.websphere.webservices.WSDL_Generation_Extra_Classpath” on page 221
- “com.ibm.ws.amm.scan.context.filter.archives” on page 221
- “com.ibm.ws.amm.scan.context.filter.packages.” on page 222
- “com.ibm.ws.application.enhancedScanning” on page 223
- “com.ibm.ws.cache.CacheConfig.alwaysSetSurrogateControlHdr” on page 223
- “com.ibm.ws.cache.CacheConfig.cascadeCachespecProperties” on page 223
- “com.ibm.ws.cache.CacheConfig.filteredStatusCodes” on page 223
- “com.ibm.ws.CacheConfig.alwaysTriggerCommandInvalidations” on page 224
- “com.ibm.ws.classloader.allowDisposedClassLoad” on page 224
- “com.ibm.ws.classloader.strict” on page 224
- “com.ibm.ws.classloader.zipFileCacheSize” on page 224
- “com.ibm.ws.el.reuseEvaluationContext” on page 225
-
- “com.ibm.ws.management.connector.soap.logClientInfo” on page 225
- “com.ibm.ws.management.repository.tempFileKeepTimeMinutes” on page 225
- “com.ibm.ws.management.repository.tempFileSweepIntervalMinutes” on page 225
- “com.ibm.ws.odr.pluginconfig.config.ASDisableNagle” on page 226
- “com.ibm.ws.odr.pluginconfig.config.AcceptAllContent” on page 226
- “com.ibm.ws.odr.pluginconfig.config.AppServerPortPreference” on page 226

- “com.ibm.ws.odr.plugincfg.config.ChunkedResponse” on page 226
- “com.ibm.ws.odr.plugincfg.config.IISDisableNagle” on page 227
- “com.ibm.ws.odr.plugincfg.config.IISPluginPriority” on page 227
- “com.ibm.ws.odr.plugincfg.config.IgnoreDNSFailures” on page 227
- “com.ibm.ws.odr.plugincfg.config.RefreshInterval” on page 227
- “com.ibm.ws.odr.plugincfg.config.ResponseChunkSize” on page 228
- “com.ibm.ws.odr.plugincfg.config.VHostMatchingCompat” on page 228
- “com.ibm.ws.odr.plugincfg.config.TrustedProxyEnable” on page 228
- “com.ibm.ws.odr.plugincfg.log.Name” on page 228
- “com.ibm.ws.odr.plugincfg.log.LogLevel” on page 228
- “com.ibm.ws.odr.plugincfg.cluster.CloneSeparatorChange” on page 229
- “com.ibm.ws.odr.plugincfg.cluster.LoadBalance” on page 229
- “com.ibm.ws.odr.plugincfg.cluster.PostSizeLimit” on page 229
- “com.ibm.ws.odr.plugincfg.cluster.RemoveSpecialHeaders” on page 229
- “com.ibm.ws.odr.plugincfg.cluster.RetryInterval” on page 230
- “com.ibm.ws.odr.plugincfg.odrIncludeStopped” on page 230
- “com.ibm.ws.odr.plugincfg.server.ConnectTimeout” on page 230
- “com.ibm.ws.odr.plugincfg.server.ExtendedHandShake” on page 230
- “com.ibm.ws.odr.plugincfg.server.MaxConnections” on page 231
- “com.ibm.ws.odr.plugincfg.cluster.WaitForContinue” on page 231
- “com.ibm.ws.odr.plugincfg.property.ESIEnable” on page 231
- “com.ibm.ws.odr.plugincfg.property.ESIMaxCacheSize” on page 231
- “com.ibm.ws.odr.plugincfg.property.ESIInvalidationMonitor” on page 231
- “com.ibm.ws.odr.plugincfg.property.https.keyring” on page 232
- “com.ibm.ws.odr.plugincfg.property.https.stashfile” on page 232
- “com.ibm.ws.odr.plugincfg.property.PluginInstallRoot” on page 232
- “com.ibm.ws.pm.checkingDBconnection” on page 232
- “com.ibm.ws.runtime.component.ResourceMgr.postBindNotify” on page 232
- “com.ibm.ws.runtime.logThreadPoolGrowth” on page 232
- “com.ibm.ws.scripting.apptimeout” on page 233
- “com.ibm.ws.sib.webservices.useSOAPJMSTextMessages” on page 233
- “com.ibm.ws.use602RequiredAttrCompatibility” on page 233
- “com.ibm.ws.webservices.allowNoSOAPActionHeader” on page 233
- “com.ibm.ws.webservices.allowStatusCode202OneWay” on page 234
- “com.ibm.ws.webservices.appendRootCauseToWSF” on page 234
- “com.ibm.ws.webservices.contentTransferEncoding” on page 234
- “com.ibm.ws.webservices.disableSOAPElementLazyParse” on page 234
- “com.ibm.ws.webservices.engine.transport.jms.propagateOneWaySystemExceptions” on page 235
- “com.ibm.ws.webservices.HttpRedirectWithProxy” on page 235
- “com.ibm.ws.webservices.ignoreUnknownElements” on page 235
- “com.ibm.ws.webservices.jaxrpc.parse.tolerate.invalid.namespace” on page 235
- “com.ibm.ws.webservices.resolveXMLSchemaDTD” on page 236
- “com.ibm.ws.webservices.searchForAppServer” on page 236
- “com.ibm.ws.webservices.serialize.2DimArray.asArrays” on page 236
- “com.ibm.ws.webservices.serializeDetailElementUsingDefaultNamespace” on page 237

- “com.ibm.ws.webservices.suppressHttpRequestPortSuffix” on page 238
- “com.ibm.ws.websvcs.attachments.sizethreshold” on page 238
- “com.ibm.ws.websvcs.suppressHttpRequestPortSuffix” on page 238
- “com.ibm.ws.websvcs.unmanaged.client.dontUseOverriddenEndpointUri” on page 238
- “com.ibm.ws.ws.wsba.protocolmessages.twoway” on page 239
- “com.ibm.wsspi.amm.merge.ignoreValidationExceptions” on page 239
- “com.ibm.xml.xlsp.jaxb.opti.level” on page 239
- “config_consistency_check” on page 240
- “deactivateWildcardURIMapping” on page 240
- “disableWSAddressCaching” on page 241
- “DRS_THREADPOOL_ISGROWABLE” on page 241
- “DRS_THREADPOOL_MINSIZE” on page 241
- “DRS_THREADPOOL_MAXSIZE” on page 241
- “java.util.logging.configureByLoggingPropertiesFile” on page 241
- “jaxws.payload.highFidelity” on page 242
- “jaxws.provider.interpretNullAsOneway” on page 242
- “jaxws.runtime.restrictStaticWebmethod” on page 243
- “jaxws.soapfault.local.exceptions.disable” on page 243
- “ODCClearMessageAge” on page 243
- “org.eclipse.jst.j2ee.commonarchivecore.FILTERBINARIES” on page 244
- “sizeThreshold” on page 244
- “threadpool.maxsize” on page 244
- “webservices.unify.faults” on page 245

allowDeployerRoleGenPluginCfg

Set this custom property to true to enable users with the deployer role to generate and configure the plugin-cfg.xml file. After you set and save this custom property, restart the application server.

If the custom property is missing or its value is set to false, the following situations occur when the user has the deployer role permissions:

- The generation and configuration processes fail.
- An error message is issued.

To disable this function, delete the custom property or set its value to false.

Also, you can set this custom property from the command line using the wsadmin tool and the following Jacl script:

```
#-----
# setAllowDeployer.jacl - Jacl script for setting a the allowDeployerRoleGenPluginCfg
property
# of the web server plug-in for WebSphere Application Server
#-----
# This Jacl file modifies the server.xml file for an application
server. This script is designed
# to be invoked while the AppServer is running.
#
# Here is an example of how to invoke the script:
# wsadmin -f setAllowDeployer.jacl &lt;nodeName> &lt;serverName> &lt;enableValue>;
#-----
proc printUsageAndExit {} {
```

```

    puts " "
    puts "Usage: wsadmin -f setAllowDeployer.jacl &lt;nodeName> <serverName> <boolEnable>"
    puts "Note: enableValue argument is of type boolean; valid values
are true and false."
    exit
}
if { [llength $argv] >= 3 } {
    set nodename [lindex $argv 0]
    set servername [lindex $argv 1]
    set enablevalue [lindex $argv 2]
} else {
    printUsageAndExit
}

set cellname [$AdminControl getCell]
set propName "allowDeployerRoleGenPluginCfg"
set propdesc "Allow conditional deployer role for plug-in generation
and propagation"
set required "false"

set jvm [$AdminConfig getid
/Cell:${cellname}/Node:${nodename}/Server:${servername}/JavaProcessDef:/JavaVirtualMachine:/]

$AdminConfig modify $jvm [subst {{systemProperties {{{name {$propName}}
{value {$enablevalue}}
{description {$propdesc}} {required {$required}}}}}}]
$AdminConfig save

exit 0

```

Usage:

```
wsadmin -f setAllowDeployer.jacl node_name server_name true
```

com.ibm.config.eclipse.wtp.enablejemtrim

Use this custom property to enable the pruning of intermediate DOM nodes after the XML parse of the metadata occurs for an application.

gotcha: The setting for this property should match the setting for the `com.ibm.config.eclipse.wtp.enablexmltrim` custom property. Either both of these properties should be left unset, set to false, or set to true

The default value for this property is false.

com.ibm.config.eclipse.wtp.enablexmltrim

Use this custom property to enable the sharing of JavaClass instances, and the conversion of expanded JavaClass and JavaMethod objects to lightweight proxies after they are used.

gotcha: The setting for this property should match the setting for the `com.ibm.config.eclipse.wtp.enablejemtrim` custom property. Either both of these properties should be left unset, set to false, or set to true.

The default value for this property is false.

com.ibm.config.eclipse.wtp.jem=finer

Use this custom property to generate a trace from code areas that are enabled when the `com.ibm.config.eclipse.wtp.enablejemtrim` custom property is set to true.

gotcha: This property might impact performance. Therefore, this property should only be specified if a problem occurs during the pruning of intermediate DOM nodes after the XML parse of the metadata occurs for an application, and you need to obtain additional information to diagnose the cause of that problem.

com.ibm.config.eclipse.wtp.xmltrim=finer

Use this custom property to generate a trace from code areas that are enabled when the `com.ibm.config.eclipse.wtp.enablexmltrim` custom property is set to `true`.

gotcha: This property might impact performance. Therefore, this property should only be specified if a problem occurs with the sharing of `JavaClass` instances, or the conversion of expanded `JavaClass` and `JavaMethod` objects to lightweight proxies after they are used, and you need to obtain additional information to diagnose the cause of that problem.

com.ibm.eclipse.wtp.allowRootedEntries

In previous service releases, properties files in the root directory of an enterprise archive (EAR) file are not read properly. Thus, in this service release and later, the behavior is changed so that the class path in `META-INF` and `MANIFEST.MF` files are treated as a relative URI. To revert back to the original behavior, set the `com.ibm.eclipse.wtp.allowRootedEntries` to `true`.

com.ibm.ejs.ras.writeSystemStreamsDirectlyToFile

Use this custom property to support JSR-47 customized logging to write to the `SystemOut` stream without the format of WebSphere Application Server. The format of WebSphere Application Server includes information, for example, timestamp, thread ID, and some others. An application might not want this information to appear in the `SystemOut` stream (or perhaps prefer the information to appear in a different format). To disable the format of WebSphere Application Server, set this custom property to `true`.

com.ibm.ejs.sm.server.quiesceInactiveRequestTime

Use this custom property to specify, in milliseconds, how fast I/O requests through the Object Request Broker (ORB) can be received and processed. For example, if you specify a value of 5000 for this property, the server does not attempt to shutdown until incoming requests are spaced at least 5 seconds apart. If the value specified for this property is too large, when the application server is stopped from the administrative console the following error message might be issued:

An error occurred while stopping Server1. Check the error logs for more information.

The default value is 5000 (5 seconds).

If you decide to use this custom property, you can specify it as a JVM custom property for either an application server, a node agent, or a deployment manager. It is typically set as an application server JVM custom property.

com.ibm.ejs.sm.server.quiesceTimeout

Use this custom property to specify, in seconds, the overall length of the quiesce timeout. If a request is still outstanding after this number of seconds, the server might start to shut down. For example, a value of 180 would be 3 minutes.

The default value is 180.

If you decide to use this custom property, you can specify it as a JVM custom property for either an application server, a node agent, or a deployment manager. It is typically set as an application server JVM custom property.

com.ibm.websphere.deletejspclasses

Use this property to indicate that you want to delete JavaServer Pages classes for all applications after those applications have been deleted or updated. The default value for this property is `false`.

com.ibm.websphere.deletejspclasses.delete

Use this property to indicate that you want to delete JavaServer Pages classes for all applications after those applications have been deleted, but not after they have been updated. The default value for this property is `false`.

com.ibm.websphere.deletejspclasses.update

Use this property to indicate that you want to delete JavaServer Pages classes for all applications after those applications have been updated, but not after they have been deleted. The default value for this property is `false`.

com.ibm.websphere.ejb.UseEJB61FEPScanPolicy

Use this property to control whether the product scans pre-Java EE 5 modules for additional metadata during the application installation process or during server startup. By default, these legacy EJB modules are not scanned.

The default value for this custom property is `false`.

You must set this property to `true` for each server and administrative server that requires a change in the default value.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.ejbcontainer.expandCMPCFJNDIName

The EJB container should allow for the expansion of the CMP Connection Factor JNDI Name when a user's JNDI name contains a user defined Application Server variable. The custom property, `com.ibm.websphere.ejbcontainer.expandCMPCFJNDIName`, makes it possible to expand the CMP Connection Factory JNDI Name.

If the value is **true**, which is the default, the EJB Container expands a variable when found in the CMP Connection Factory JNDI Name. If the value is set to **false**, the EJB Container does not expand a variable.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.ejbcontainer.includeRootExceptionOnRollback

Use this Enterprise JavaBeans (EJB) custom property to enable the following functionality:

- Allow the root cause of a transaction roll back to be included in a `TransactionRolledbackLocalException` if the transaction is issued by a local caller.
- Allow the root cause of a transaction roll back from a from the commit method to be included in a `TransactionRolledbackLocalException` even if the transaction is issued by a remotecaller.
- Allow Heuristic Exceptions to be returned rather than a `TransactionRolledbackLocalException`, for a local client, or a `TransactionRolledbackLocalException`, for a remote client.
- Allow a `RemoteException` to be returned from a remote EJB method even if that method is running in the context of the transaction of the call. For example, consider that EJB1, method `m1`, begins a transaction and calls EJB2, method `m2`, where `m2` causes an unhandled exception. In this case, the EJB Specification mandates that `m1` receives a `TransactionRolledbackException`. Setting this property to `true` allows a `RemoteException`, that includes any nested exceptions to be returned instead of the a `TransactionRolledbackException` even though this functionality is contrary to the EJB Specification requirement.

To enable this functionality set this property to `true`. To disable, this functionality set this property to `false`.

The default is `false`.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.jaxrpc.stub.typemapping.per.thread

Use this property to indicate whether the JAX-RPC runtime should use thread specific type mapping objects.

The JAX-RPC runtime uses a single `TypeMappingRegistry` object for all of the JAX-RPC clients. This design is intentional, and allows you to create a JAX-RPC stub and use it on multiple threads. However the singleton `TypeMappingRegistry` gets contaminated if multiple JAX-RPC Web services with different mappings are invoked concurrently. Even though this situation is uncommon, if it exists on your system, you can set the `com.ibm.websphere.jaxrpc.stub.typemapping.per.thread` custom property to `true` to indicate to the JAX-RPC runtime that it can use thread specific type mapping objects. These separate mapping objects avoid the contamination issue, and the various web service calls will succeed.

The default value for this property is `false`.

gotcha: You should not use this custom property unless you encounter a situation where the singleton `TypeMappingRegistry` gets contaminated. Enabling this property might regress applications that are dependent on access to the same JAX-RPC stub across multiple threads.

com.ibm.websphere.jaxrs.server.DisableIBMJAXRSEngine

Use this property to disable the JAX-RS integration run time from automatically processing your JAX-RS applications. The default value for this property is `false`.

Note: Setting this property to a value of `true` also disables the IBM JAX-RS runtime integration with EJB and JCDI.

com.ibm.websphere.management.application.fullupdate

Use this property to specify that when any of your applications are updated, you want the binaries directory erased and the content of the updated EAR file completely extracted.

If this property is not specified, each changed file within an updated EAR file is individually updated and synchronized in the node. This process can be time consuming for large applications if a large number of files change.

Setting the `com.ibm.websphere.management.application.fullupdate` property to:

- `true` specifies that, when any of your applications are updated, you want the binaries directory erased and the content of the updated EAR file completely extracted.
- `false` specifies that, when any of your applications are updated, you only want the changed files within that EAR file updated on the node and then synchronized.

gotcha: Use the `com.ibm.websphere.management.application.fullupdate.application_name` property if you only want to do a full replacement for a specific application instead of all of your applications.

com.ibm.websphere.management.application.fullupdate.application_name

Use this property to specify that when the specified application is updated, you want the binaries directory for that application erased and the content of the updated EAR file completely extracted.

If this property is not specified, each changed file within the updated EAR file for the specified application is individually updated and synchronized in the node. This process can be time consuming for large applications if a large number of files change.

Setting the `com.ibm.websphere.management.application.fullupdate.application_name` property to:

- `true` specifies that when the specified application is updated, you want the binaries directory erased and the content of the updated EAR file completely extracted.

- `false` that when the specified application is updated, you only want the changed files updated on the node and then synchronized.

gotcha: Use the `com.ibm.websphere.management.application.fullupdate` property if you want the binaries directory erased and the content of the updated EAR file completely extracted whenever any of your applications are updated.

com.ibm.websphere.management.application.keepExistingSharedLibraries

Use this property to specify how shared library mappings are handled during application updates.

When this property is set to `false`, then the shared libraries specified during the application update operation should replace the original shared library settings. `False` is the default setting.

When this property is set to `true`, after an application is updated, the application and module configurations include the original shared library settings in addition to those that are specified during the update operation.

com.ibm.websphere.management.application.persistWebContext

Use this property to specify whether the context root and virtual host information for web modules is persisted in the `deployment.xml` file. If this property is not specified, application deployment has to rely on annotation processing to read the context root and virtual host information, which impacts the performance of application deployment

When this property is set to `true`, the context root and virtual host information for web modules is persisted in the `deployment.xml` file, the persisted data is used for application deployment validation, which improves the performance of application deployment.

The default value is `false`, which means that the context root and virtual host information for web modules is not persisted in the `deployment.xml` file.

com.ibm.websphere.management.application.sync.recycleappsv5

Use this property to specify that you want your application recycling behavior to work the same way as this behavior worked in versions previous to Version 6.x of the product.

In Version 6.x and higher, after an application update or edit operation occurs, depending on which files are modified, either the application or its modules are automatically recycled. This recycling process occurs for all application configuration file changes, and all non-static file changes.

However, in versions previous to Version 6.x of the product, an application is recycled only if the Enterprise Archive (EAR) file itself is updated, or if the binaries URL attribute changes. An application is not recycled if there is a change to the application configuration file.

Setting the `com.ibm.websphere.management.application.sync.recycleappsv5` property to:

- `true` specifies that you want your application recycling behavior to work the same way as this behavior worked in versions previous to Version 6.x of the product.
- `false` specifies that you want your application recycling behavior to work according to the Version 6.x and higher behavior schema.

The default value for this custom property is `false`.

com.ibm.websphere.management.application.updateClusterTask.serverStopWaitTimeout

Use this property to specify, in seconds how long the deployment manager waits for a server to stop completely in the `$AdminTask updateAppOnCluster` task. By default, the deployment manager waits for 60 seconds. The amount of time that you specify for this property should be greater than the longest amount of time that it takes to stop a server in the cluster.

This property can only be specified if you are using Version 7.0.0.1 or higher.

gotcha: This property is only valid if it is specified for a deployment manager.

com.ibm.websphere.management.application.updatesync.appExpansionTimeout

Use the property to specify how long the deployment manager waits to start an application server following an application update. This wait time enables the binaries for the application to be expanded to their directories after the update process completes. The amount of time that you specify for this property should be the maximum amount of time that any of the applications that reside in a node, take to fully expand their binaries.

By default, the rollout update function waits for 60 seconds, for each application expansion to occur following an update to one or more applications. Because the rollout function can be used to update multiple applications at the same time, the default value for this property is $n \times 60$ seconds, where n is the number of applications that are being updated.

The default wait time might not be sufficient for larger applications. If, after your applications are updated, one or more of these applications do not start when the server starts, you might have to specify a longer length of time for the rollout update function to wait before starting the server.

gotcha: This property is only valid if it is specified for a deployment manager.

com.ibm.websphere.management.configservice.validatePropNames

Use this property to specify whether to enforce character restrictions for custom property names, and for the name value of Property and J2EEResourceProperty configuration objects in wsadmin commands.

You can use one of the following methods to turn off character validation for custom property names, and the name value of Property and J2EEResourceProperty configuration objects in wsadmin commands.

- Set the `com.ibm.websphere.management.configservice.validatePropNames` Java system property to `false` in the Java virtual machine (JVM) for the deployment manager server.
- Set the `com.ibm.websphere.management.configservice.validatePropNames` property using the **-javaoption** parameter when you use the wsadmin tooling in the local mode.

```
wsadmin -conntype none -javaoption  
"-Dcom.ibm.websphere.management.configservice.validatePropNames=false"
```

com.ibm.websphere.management.processEmbeddedConfigGlobal

Use this property to globally enable or disable processing of the embedded configuration of enhanced application Enterprise Archive (EAR) files during deployment. An enhanced EAR file results when you export an installed application.

This custom property overrides globally the default setting for the **Process embedded configuration** (`-processEmbeddedConfig`) option. By default, **Process embedded configuration** is set to `true` (selected) for enhanced EAR files and `false` (deselected) for all other EAR files. The **Process embedded configuration** setting determines the directory to which the product expands an enhanced EAR file during deployment of the enhanced EAR file. If you exported an application from a cell other than the current cell and did not specify the `$(CELL)` variable for **Directory to install application** when first installing the application, setting **Process embedded configuration** to `false` during deployment of an enhanced EAR file extracts the enhanced EAR file in the `app_server_root/profiles/installedApps/current_cell_name` directory. Otherwise, if **Process embedded configuration** is set to `true`, the enhanced EAR file is expanded in the `app_server_root/profiles/installedApps/original_cell_name` directory, where `original_cell_name` is the cell on which the application was first installed. If you specified the `$(CELL)` variable for **Directory to install application** when you first installed the application, installation expands the enhanced EAR file in the `app_server_root/profiles/installedApps/current_cell_name` directory.

When this `processEmbeddedConfigGlobal` custom property is set to `false`, the product does not process the embedded configuration of any application, including enhanced EAR files, during deployment. After you

set `processEmbeddedConfigGlobal` to `false`, the product does not process the embedded configuration of enhanced EAR files. However, when deploying an individual enhanced EAR file, you can override this false setting by explicitly setting **Process embedded configuration** to `true`.

When this `processEmbeddedConfigGlobal` custom property is set to `true`, the product processes the embedded configuration of enhanced EAR files.

Regardless of whether this `processEmbeddedConfigGlobal` custom property is set to `true` or `false`, the product deploys applications that do not have embedded configurations as usual. The setting has no effect on deployment.

com.ibm.websphere.metadata.ignoreDuplicateRefBindingsInWebModul

Use this property to control whether the JVM ignores instances of duplicate reference bindings in the DTD file for a web module in a Java 2 Platform, Enterprise Edition (J2EE) version 1.3 application. Typically a `MetaDataException` occurs if the DTD file for a web module in a Java 2 Platform, Enterprise Edition (J2EE) version 1.3 application contains duplicate references.

The standards for the DTD file for a web module specifically states that the reference bindings must have a unique name fields. Therefore, an application that contains a web module that includes duplicate reference bindings is technically a non-compliant application.

Although the standards for the DTD file for a web module forbids a user from defining duplicate reference bindings, the JVMs in versions of the product that preceded 7.0 tolerate duplicate reference bindings. If you have DTD files for web modules in Java 2 Platform, Enterprise Edition (J2EE) version 1.3 applications that contain duplicate reference bindings, you can either remove the duplicate reference, or add this property to your JVM configuration settings, and set the property to `true`.

com.ibm.websphere.network.useMultiHome

Use this property in a multihomed environment to indicate on which IP addresses the application server listens. In a multihomed environment, there is normally a specific IP address that the application server is restricted to listening on for Discovery and SOAP messages. Setting the `com.ibm.websphere.network.useMultiHome` property to:

- `true` specifies that the product listens on all IP addresses on the host for Discovery and SOAP messages.
- `false` specifies that the product only listens on the configured host name for Discovery and SOAP messages. If you set this property to `false`, you should have a host name configured on the product that resolves to a specific IP address.
- `null` specifies that the product only listens 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. You must restart the server before these changes take effect.

com.ibm.websphere.sib.webservices.useTypeSoapArray

You can pass messages directly to a bus destination by overriding the JAX-RPC client binding namespace and endpoint address. However:

- The default RPC-encoded web services string array message that is generated might not interoperate successfully with some target service providers.
- The string array message produced is not exactly the same as the standard JAX-RPC equivalent, which can interoperate successfully.

Here are examples of the two different messages:

- Service integration bus message:

```
<partname env:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/ xsi:type='ns1:ArrayOf_xsd_string'>
  <item xsi:type='xsd:anySimpleType'>namevalue</item>
</partname>
```

- **JAX-RPC client message:**

```
<partname xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]">
  <item>namevalue</item>
</partname>
```

Set this property to true to modify the default behavior and send a string array message that is fully compatible with standard JAX-RPC. Setting this property modifies the default behavior for all outbound JMS web services invocations sent from the service integration bus.

com.ibm.websphere.webservices.attachment.tempfile.expiration

Use this property to indicate, in seconds, an expiration time for an attachment on a JAX-WS or Service Component Architecture (SCA) client or service. If an attachment is not accessed for a period of time greater than the expiration time, the web service runtime is allowed to delete the attachment.

The JAX-RPC programming model allows access to attachments from incoming Web service messages. The attachment might be accessed immediately, or might be stored for later processing. Therefore, the memory associated with the attachment might persist much longer than the lifetime of the Web service interaction. Because there is no precise length of time after which the Web service runtime can safely free the attachment.

For small attachments, the memory is eventually freed by the Java garbage collector.

For large attachments, the JAX-RPC runtime stores the attachment data in a temporary file, thereby allowing the runtime to process extremely large attachments without consuming memory. If the application does not access the attachment, or if the application does not adequately close the data handler associated with the attachment, the large temporary file is not freed. Over time, these temporary attachment files might accumulate on the file system if no expiration time is specified for these files.

bprac: A setting of 600 is recommended if you need to specify an expiration time for these attachments. The default setting for this custom property is 0 seconds, which indicates that there is no expiration time for these attachments.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.attachments.maxMemCacheSize

Use this property to specify, in kilobytes, the maximum size of an attachment on the JAX-RPC client or service that can be written to memory. For example, if your web service needs to send 20 MB attachments, set the property to 20480.

When determining a value for this property, remember that the larger the maximum cache size, the more impact there is on performance, and, potentially, to the Java heap.

If you do not specify a value for this property, the maximum memory that is used to cache attachments is 32 KB, which is the default value for this property.

Note: To specify the maximum size of an attachment on the JAX-WS client or service, see the `com.ibm.ws.websvcs.attachments.sizethreshold` custom property.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.DisableIBMJAXWSEngine

Use this property to turn off web services annotation scanning at the server level. By default, web services annotation scanning is enabled at the server level.

To turn off annotation scanning at the application level, set the `DisableIBMJAXWSEngine` property in the `META-INF/MANIFEST.MF` of a WAR file or EJB module to `true`.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.http.OneWayConnectionRecycleTime

Use this property to specify, in seconds, how long the web services engine should wait before reusing a one-way connection. When a one-way connection is reused too quickly, a web service operation might fail on the client because of a timeout problem, such as a `SocketTimeoutException`.

When a value is specified for this property, one-way connections are not reset until the specified number of seconds elapses, starting from when the request is sent.

By default, this property is not set and one-way connections are reset immediately after the request is sent.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.http.waitingThreadsThreshold

Use this property to specify how many waiting connection requests are tolerated before releasing soft connections. A soft connection occurs when a client engine maintains connection objects for some hosts after the connection is closed. By default, after five threads are waiting for connections, the client engine releases the soft connections.

Note: If all of the connections are being used, the custom property does not have an impact. In this situation, you can increase the maximum connection limit, the maximum number of threads, or both.

The default value for this custom property is 5.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.jaxrpc.client.publishwsdl

Specifies whether a WSDL file is published for a client web module. When this property is set to `true`, if an application contains a client web module, a WSDL file might be published for that client. If you do not want WSDL files published for your client applications, set this property to `false`.

The default value of this property is `true`.

gotcha:

- WSDL file publication is not available for JAX-RPC applications that only contain a client, .
- This property cannot be used for JAX-WS applications.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.soap.enable.legacy.get.behavior

Note: Starting in WebSphere Application Server Version 8, the SOAP with Attachments API for Java (SAAJ) methods `SOAPMessage.getSOAPHeader` and `getSOAPBody` now throw a `SOAPException` if there is no corresponding element in the message. Previously these methods would return a null if there was no corresponding element in the message. A System property is provided to revert the behavior to return null rather than throw an exception. The property is `com.ibm.websphere.webservices.soap.enable.legacy.get.behavior`. The default value of the property is `null` which is interpreted as `false`. To revert the behavior to returning a null, set the property to the String value `true`. Note that the previous behavior of returning null is not compliant with the SAAJ specification.

com.ibm.websphere.webservices.tempAttachDir

Use this property to specify the location on a storage device where you want the web services runtime to cache a copy of any attachment, that is greater than 32KB in size, that is being sent or received as part of a SOAP message.

For performance reasons, the web services runtime caches a temporary copy of any SOAP message attachment that is greater than 32KB in size. If you do not specify a value for this property, the cached copy of the attachment is typically sent to the default temporary directory for your operating system.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.transport.jms.messageType

Use this property to control the JMS message type that is used by the web services engine for SOAP over JMS components when sending request and response messages. To specify a JMS `BytesMessage` (`javax.jms.BytesMessage`) object, set the property to `BYTES` to indicate the body of the message is binary data. To specify a JMS `TextMessage` (`javax.jms.TextMessage`) object, set the property to `TEXT` to indicate the body of the message is string data.

The default value for this custom property is `BYTES`.

To learn more about the SOAP over JMS message types, see the configuring SOAP over JMS message types information.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.transport.OPTIMIZE_HTTP_HEADERS

Prior to Version 8, a JAX-WS client application for WebSphere Application Server might send a `SAVE_CONNECTION` HTTP header in a SOAP message. This additional header ensures that proper processing occurs by the application server that is hosting the JAX-WS web service. However, this `SAVE_CONNECTION` header and the additional processing is not necessary if the application server for the client and the application server host for the web service are both using WebSphere Application Server Version 7.0 Fix Pack 3 or later.

You can set the `com.ibm.websphere.webservices.transport.OPTIMIZE_HTTP_HEADERS` custom property to `false` to enable the `SAVE_CONNECTION` header to ensure proper processing by older application server levels. By default, this custom property is set to `true`, which disables the JAX-WS client from sending the `SAVE_CONNECTION` header. If you need to change this default behavior, set the custom property to `false` on the application server that is hosting the JAX-WS client application.

Important: You must verify that the application server for the client and application server host for the web service are both using Version 7.0 Fix Pack 3 or later.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.transport.ssl.loadFromPolicyBinding

Use this property to control whether JAX-WS applications use SSL transport bindings or the system default SSL settings when the client is a managed client, and the client and the server are in different application servers.

When you create an SSL binding, this property is automatically added to the bindings file, and set to true. This setting enables SSL transport bindings to be used for JAX-WS applications when the client is a managed client, and the client and the server are in different application servers. If no bindings are attached to your JAX-WS application, set this property to false.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.UseWSFEP61ScanPolicy

Use this property to control whether the product scans WAR 2.4 and earlier modules for JAXWS components and semi-managed service clients. By default, these legacy WAR modules are only scans for semi-managed service clients.

The default value for this custom property is false.

You must set this property to true for each server and administrative server that requires a change in the default value.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.websphere.webservices.WSDL_Generation_Extra_Classpath

Use this property to set the location of the shared class files. The wsgen command-line tool generates the necessary artifacts that are required for Java API for XML Web Services (JAX-WS) applications when they start from Java code. However, the wsgen command-line tool might not locate the necessary class files and append the following error messages to the log file:

```
Caused by: java.lang.NoClassDefFoundError
...
at com.ibm.ws.websvcs.wsd1.WASWSDLGenerator.wsgen(WASWSDLGenerator.java:521)
at com.ibm.ws.websvcs.wsd1.WASWSDLGenerator.generateWsd1(WASWSDLGenerator.java:183)
```

Use this property to provide the fully qualified location to the missing class files. With this custom property, you can provide fully qualified paths to multiple Java archives (JAR) and directories and separate them using a semicolon (;).

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.amm.scan.context.filter.archives

Use this property to provide a list of archives, or utility JAR files, that do not contain annotations. Archives or utility JAR files specified for this property are not be scanned for annotations.

When a Java Platform, Enterprise Edition (Java EE) 5 or 6 application is deployed or updated, the Annotations Metadata Manager (AMM) facility scans all of the annotation metadata. This scanning process can negatively affect the amount of time required to deploy an application. If the application includes archives or utilities that do not contain annotations, you can list these archives and utilities as the value for this property. If the application includes Java Packages that do not contain annotations, you can list them as the value for the Ignore-Scanning-Packages property.

The values specified for this properties are case sensitive, and must be expressed as a single string with a comma followed by a space used to separate the names of the archives or utility JAR files. Wildcards and REGEX expressions are not permitted.

As an alternative to using this custom property, you can add the Ignore-Scanning-Archives property to one of the following files or modules, and specify the archives and utilities that you do not want scanned as the value of that property:

- The amm.filter.properties file that is located in the was_home/properties directory.
- The amm.filter.properties file that is located in the profile_home/properties directory
- The manifest file of an application, META-INF/MANIFEST.MF
- The manifest of a web or Enterprise JavaBeans (EJB) module within an application

Values specified in the amm.filter.properties files are merged with those found in this custom property to form a server scoped set of filters. This merged set of filters applies to all of the applications that are deployed on that server.

Values specified in the manifest file of an application are merged with the server scoped set of filters to form a module scoped superset that applies to all modules within that application.

Values specified in the manifest file of a web or Enterprise JavaBeans (EJB) module are merged with the module scoped set of filters. This merged set of filters only applies to that module.

gotcha: Exercise caution if you update a manifest file. Manifest files have line length limitations, and other constraints that must be adhered to.

Example:

```
Ignore-Scanning-Archives : ant.jar, avalon-framework-4.2.0.jar, axis.jar, CICS.jar, xerces.jar
```

com.ibm.ws.amm.scan.context.filter.packages.

Use this property to provide a list of Java Packages that do not contain annotations. The Java classes specified for this property are not scanned for annotations.

When a Java Platform, Enterprise Edition (Java EE) 5 or 6 application is deployed or updated, the Annotations Metadata Manager (AMM) facility scans all of the annotation metadata. This scanning process can negatively affect the amount of time required to deploy an application. If the application includes Java Packages that do not contain annotations, you can list them as the value for this property. If the application includes archives or utilities that do not contain annotations, you can list them as the value for the Ignore-Scanning-Archives property.

The value specified for this property are case sensitive and must be expressed as a single string with a comma followed by a space used to separate the names of the Java Packages. Wildcards and REGEX expressions are not permitted.

As an alternative to using this custom property, you can add the Ignore-Scanning-Packages property to one of the following files or modules, and specify the archives and utilities that you do not want scanned as the value of that property:

- The amm.filter.properties file that is located in the was_home/properties directory
- The amm.filter.properties file that is located in the profile_home/properties directory
- The manifest file of an application, META-INF/MANIFEST.MF
- The manifest of a web or Enterprise JavaBeans (EJB) module within an application

Values specified in the amm.filter.properties files are merged with those found in this custom property to form a server scoped set of filters. This merged set of filters applies to all of the applications that are deployed on that server.

Values specified in the manifest file of an application are merged with the server scoped set of filters to form a module scoped superset that applies to all modules within that application.

Values specified in the manifest file of a web or Enterprise JavaBeans (EJB) module are merged with the module scoped set of filters. This merged set of filters only applies to that module.

gotcha: The following example is properties file centric and cannot be used as is for a manifest file. Manifest files have a 72 byte line length limit, as well as other constraints that must be adhered to.

Example:

```
Ignore-Scanning-Packages : org.apache.avalon, org.apache.batik, org.apache.commons
```

com.ibm.ws.application.enhancedScanning

Use this property to disable several optimizations that decreases the time to deploy and start enterprise applications. The optimizations primarily involve Java Platform, Enterprise Edition (Java EE 5)-enabled applications. When you set this property to false, the following updates are disabled:

- A new cache for modules files
- A new cache for module classloading
- Alternate code paths for annotation processing.

If you set this property to false, you might experience decreases in performance. Thus, by default, this property value is set to true.

com.ibm.ws.cache.CacheConfig.alwaysSetSurrogateControlHdr

Use this property to force the surrogate-control header from the dynamic cache service to always be set on the response. The surrogate-control header contains the metadata that the Edge Side Include (ESI) processing needs to correctly generate, and invalidate the cached content in the ESI cache.

The default value is false, which means that the surrogate-control header might not be set on the response.

If you decide to use this custom property, specify it as an application server JVM custom property unless otherwise indicated within the context of a specific task.

com.ibm.ws.cache.CacheConfig.cascadeCachespecProperties

Use this property to enable child pages or fragments to inherit the cascade of save-attributes, and store-cookies properties from their parent pages or fragments.. The default value is false.

The default behavior of the dynamic cache service is to store the request attributes for a child page or fragment, if not explicitly overridden in the cache specification. An application server can run into an Out-Of-Memory condition in scenarios where these request attributes get too large. If the attributes saved by default are not serializable, then the disk offload of these cache entries results in java.io.NotSerializableExceptions.

If you decide to use this custom property, specify it as an application server JVM custom property unless otherwise indicated within the context of a specific task.

com.ibm.ws.cache.CacheConfig.filteredStatusCodes

Use this property to indicate error situations in which you do not want the dynamic cache service to cache the servlet output.

The value specified for this property is a space delimited list of HTTP response error codes. If the status code returned from a cache miss matches one of the listed response error codes, the dynamic cache service does not cache the data that was obtained in response to an HTTP request.

If you decide to use this custom property, specify it as an application server JVM custom property unless otherwise indicated within the context of a specific task.

com.ibm.ws.CacheConfig.alwaysTriggerCommandInvalidations

Use this property to ensure that command invalidations are triggered regardless of the skipCache attribute.

When a request object contains the <previewRequest> attribute the dynamic cache sets the skipCache attribute to true. When the skipCache attribute is true, commands are not always invalidated. Set the com.ibm.ws.CacheConfig.alwaysTriggerCommandInvalidations custom property to true to ensure that command invalidations are triggered regardless of the skipCache attribute. When you set this custom property, it affects all cache instances. The default value for this property is false.

This custom property is set on the application server level only.

If you decide to use this custom property, specify it as an application server JVM custom property unless otherwise indicated within the context of a specific task.

com.ibm.ws.classloader.allowDisposedClassLoad

Use this property to specify whether the JVM should fully dispose of an application class loader when the application is stopped. If the JVM does not fully dispose of an application class loader, classes can still be loaded from that class loader.

When this property is set to a value of true, the JVM does not fully dispose of an application class loader when the application is stopped.

The default value for this property is false.

transition: The default value for this property was true in previous versions of the product.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.classloader.strict

Use this property to enable the WebSphere Application Server application class loader to provide access to the META-INF directory through a getResources call even if the META-INF directory path does not include a trailing slash

The WebSphere Application Server application class loader does not, by default, provide access to the META-INF directory through a getResources call unless a trailing slash is specified at the end of the META-INF directory path. If you want the WebSphere Application Server application class loader to provide access to the META-INF directory through a getResources call even if a trailing slash is not specified at the end of the META-INF directory path, set this property to true.

The default value for this property is false.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.classloader.zipFileCacheSize

Use this property to specify the maximum number of application JAR files that can be held open for resource and class loading. Reducing the number of times JAR files must be opened, improves the performance of applications that are resource or class loading intensive.

When the specified limit of open JAR files is reached, the class loader starts to close and remove JAR files based on the last time they were accessed. The most recently accessed JAR files are kept open. The value specified for this property should be based on the total number of application JAR files that are frequently accessed.

The default value for this property is 8. Specifying a value of 0 disables the cache and prevents application JAR files from being held open.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.el.reuseEvaluationContext

Use this property to indicate that the same EvaluationContext object can be reused on a per thread basis.

Typically, during the evaluation of expressions the Unified EL code creates a new `org.apache.el.lang.EvaluationContext` object for each call that is made. Because these objects are subsequently made available for garbage collection, as the number of objects created increases, memory consumption and garbage collection also increases. Setting the `com.ibm.ws.el.reuseEvaluationContext` property to `true` enables the same EvaluationContext object to be reused on a per thread basis., thereby decreasing memory consumption and the amount of garbage collection that needs to occur.

The default value is `false`.

com.ibm.ws.management.connector.soap.logClientInfo

Use this property to indicate whether you want to log the host, port, and username of SOAP client requests. When this property is set to `true`, SOAP client details are logged in `SystemOut.log`. These details are also added to `trace.log` if the trace level for the SOAP connector is set to `all`.

The default value for this property is `false`.

com.ibm.ws.management.repository.tempFileKeepTimeMinutes

Use this property to specify, in minutes, how long a file is kept in the configuration repository temporary directory before the configuration repository temporary directory cleanup task can delete that file from the directory. The default value for this property is 1440 minutes, which is equal to 24 hours. In previous versions of the product, a file was kept for 60 minutes.

The default value is typically sufficient for performing needed cleanup without deleting files that are in use. However, there might be situations where you need to specify a larger, or smaller value. You can specify a minimum value of 60 minutes for this property. However, it is recommended that you specify a value that is equivalent to several hours to account for situations where very large files are being transferred or synchronized, or where a network is slow, and file transfer operations are taking a long time. In these situations, if too short a time period is specified, it is possible for a file to be deleted while it is still being transferred.

If the `com.ibm.ws.management.repository.tempFileSweepIntervalMinutes` property is set to 0, the cleanup function is disabled, and any files left behind after a server process failure, remain in the configuration repository temporary directory until they are manually removed, or the cleanup function is enabled.

gotcha: If an invalid value is specified for this property, the default value is used.

com.ibm.ws.management.repository.tempFileSweepIntervalMinutes

Use this property to specify, in minutes, how frequently the configuration repository temporary directory cleanup task runs. This task removes files from the configuration repository temporary directory that were not properly removed because of a server process failure.

The cleanup task always runs when the server starts, and then again after the time length specified for this property expires. The default value for this property is 720 minutes, which is equivalent to 12 hours. This length of time is typically sufficient for the configuration repository temporary directory cleanup task to successfully complete the cleanup process. You can disable this cleanup function by setting this property to 0.

In previous versions of the product, the cleanup task ran when the server started, and then ran again every 30 minutes.

gotcha: If an invalid value is specified for this property, the default value is used.

com.ibm.ws.odr.plugincfg.config.ASDisableNagle

Use this property to specify whether you want to disable the Nagle algorithm for the connection between the plug-in and the proxy server.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `false`, which means that the Nagle algorithm is enabled for the connection between the plug-in and the proxy server.

com.ibm.ws.odr.plugincfg.config.AcceptAllContent

Use this property to specify whether you 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.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `false`.

com.ibm.ws.odr.plugincfg.config.AppServerPortPreference

Use this property to specify which port number is used to build URI's for a `sendRedirect`.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `HostHeader`.

com.ibm.ws.odr.plugincfg.config.ChunkedResponse

Use this property to specify whether the plug-in groups the response to the client when a Transfer-Encoding : Chunked response header is present in the response.

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.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `false`.

com.ibm.ws.odr.plugincfg.config.IISDisableNagle

Use this property to specify whether you want to disable the nagle algorithm.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `false`.

com.ibm.ws.odr.plugincfg.config.IISPluginPriority

Use this property to specify the priority in which the Web server loads the plug-in. You can specify one of the following values for this attribute:

- High
- Medium
- Low

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `High`.

com.ibm.ws.odr.plugincfg.config.IgnoreDNSFailures

Use this property to specify whether the plug-in is to ignore DNS failures within a configuration when started.

When this property is set to `true`, the plug-in ignores DNS failures within a configuration and starts successfully if at least one server in each `ServerCluster` resolves the host name. Any server for which the host name is not resolved is marked unavailable for the life of the configuration. The host name is not resolved later 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 continues initializing instead of the Web server not starting.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `false`.

com.ibm.ws.odr.plugincfg.config.RefreshInterval

Use this property to specify, in seconds, how frequently the plug-in should check the configuration file for updates or changes. The plug-in checks the file for any modifications that occur since the plug-in configuration was loaded.

In a development environment where frequent changes occur, set the time interval to less than 60 seconds.

In a production environment, you should set a higher value than the default value, because updates to the configuration do not occur as frequently.

If the plug-in reload is not successful, the plug-in log file contains an error message, and the previous configuration is used until the plug-in configuration file successfully reloads. Refer to the plug-in log file for more information if an error occurs.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `60`.

com.ibm.ws.odr.plugincfg.config.ResponseChunkSize

Use this property to specify, in kilobytes, the maximum chunk size the plug-in should use when reading the response body. For example, `Config ResponseChunkSize="N">`, where *N* equals the chunk size.

By default, the plug-in reads the response body in 64k chunks until all of the response data is read. This process might cause a performance problem for requests where the response body contains large amounts of data. 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* is used to read the response body.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is 64.

com.ibm.ws.odr.plugincfg.config.VHostMatchingCompat

Use this property to specify whether the plug-in should use the port number for virtual host matching. The following values can be specified:

- True for physically matching by using the port number for which the request is received.
- False for logically matching by using the port number contained in the host header.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is false.

com.ibm.ws.odr.plugincfg.config.TrustedProxyEnable

Use this property to specify whether the plug-in is to allow the inclusion of trusted proxies. The following values can be specified:

- True if you want to allow the inclusion of trusted proxies.
- False if you do not want to allow the inclusion of trusted proxies.

The trusted proxies are collected from the defined trusted security proxies.

This property is only valid for a proxy server, and applies to the Config element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is false.

com.ibm.ws.odr.plugincfg.log.Name

Use this property to specify the fully qualified path to the log file to which the plug-in writes error messages.

This property is only valid for a proxy server, and applies to the Log element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `profileRoot/logs/http_plugin.log`.

com.ibm.ws.odr.plugincfg.log.LogLevel

Use this property to specify the level of detail of the log messages that the plug-in writes 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.

This property is only valid for a proxy server, and applies to the Log element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is Error.

gotcha: A lot of messages are logged at the trace level, which can cause the file system to fill up very quickly. Never use a trace setting in a normally functioning environment as it adversely affects performance.

com.ibm.ws.odr.plugincfg.cluster.CloneSeparatorChange

Use this property to indicate to the plug-in that the plus character (+) can be used as the clone separator.

Some pervasive devices cannot handle the colon character (:) that is used to separate clone IDs in conjunction with session affinity.

gotcha: If you use this custom property, you must change the proxy server configurations such that the proxy server separates clone IDs with the plus character instead the colon character.

This property is only valid for a proxy server, and applies to the ServerCluster element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is false.

com.ibm.ws.odr.plugincfg.cluster.LoadBalance

Use this property to specify the appropriate load balancing option: Round Robin or Random.

The Round Robin implementation has a random starting point. The first proxy server is picked randomly. Round Robin is then used to pick proxy 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 proxy server.

The Random implementation also has a random starting point. However with this implementation all subsequent proxy servers are also randomly selected. Therefore, the same proxy server might get selected repeatedly while other proxy servers remain idle.

The default value is Round Robin.

com.ibm.ws.odr.plugincfg.cluster.PostSizeLimit

Use this property to specify, in bytes, the maximum number of bytes of request content that the plug-in is allowed to attempt to send to a server. If a request is received that is greater than the specified value, the plug-in ends the request

This property is only valid for a proxy server, and applies to the ServerCluster element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is -1, which indicates that there is no limit to the size of a request.

com.ibm.ws.odr.plugincfg.cluster.RemoveSpecialHeaders

Use this property to whether the plug-in is to add special headers to a request before it is forwarded to the server. These headers store information about the request that the application then uses. By default, the plug-in removes these headers from incoming requests before adding the required headers.

If you set this property to false, you introduce a potential security exposure headers from incoming requests are not removed.

This property is only valid for a proxy server, and applies to the `ServerCluster` element in the `plugin-cfg.xml` file that the proxy server automatically generates.

The default value is `true`.

com.ibm.ws.odr.plugincfg.cluster.RetryInterval

Use this property to specify, in seconds, the amount of time that elapses between when a proxy server is marked down and when the plug-in reattempts to make a connection.

This property is only valid for a proxy server, and applies to the `ServerCluster` element in the `plugin-cfg.xml` file that the proxy server automatically generates.

The default value is `60`.

com.ibm.ws.odr.plugincfg.odrIncludeStopped

Use this property to specify whether the plug-in is to allow the inclusion of stopped proxy servers. The following values can be specified:

- `True` if you want to allow the inclusion of stopped proxy servers.
- `False` if you do not want to allow the inclusion of stopped proxy servers.

This property is only valid for a proxy server.

The default value is `false`.

com.ibm.ws.odr.plugincfg.server.ConnectTimeout

Use this property to specify, in seconds, the amount of time the plug-in waits for a successful connection

Specifying a value for this property enables the plug-in to perform non-blocking connections with the proxy server. Such connections are beneficial when the plug-in is unable to contact the destination to determine if the port is available or unavailable.

When a value greater than `0` is specified, and a connection does not occur after that time interval elapses, the plug-in marks the proxy server unavailable, and proceeds with one of the other proxy servers defined in the cluster.

If no value is specified for this property, the plug-in performs a blocking connect in which the plug-in waits until an operating system times out and allows the plug-in to mark the proxy server unavailable.

This property is only valid for a proxy server, and applies to the `Server` element in the `plugin-cfg.xml` file that the proxy server automatically generates.

The default value is `0`.

com.ibm.ws.odr.plugincfg.server.ExtendedHandShake

Use this property to indicate to the plug-in that it must ensure the availability of a proxy server before sending a request to that proxy server.

Typically, the plug-in marks a proxy server as stopped when a `connect()` ends. However, when a proxy firewall is between the plug-in and the proxy server, the `connect()` succeeds, even though the back-end proxy server is stopped. This situation causes the plug-in to not failover correctly to other proxy servers.

This property is only valid for a proxy server, and applies to the `Server` element in the `plugin-cfg.xml` file that the proxy server automatically generates.

The default value is false.

com.ibm.ws.odr.plugincfg.server.MaxConnections

Use this property to specify the maximum number of pending connections to a proxy server that can flow through a Web server process at any point in time.

This property is only valid for a proxy server, and applies to the Server element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is -1, which indicates that there is no maximum number for the number of pending connections to a proxy server that can flow through a Web server process at any point in time.

com.ibm.ws.odr.plugincfg.cluster.WaitForContinue

Use this property to specify whether to use the HTTP 1.1 100 Continue support before sending the request content to the proxy server.

Typically, the plug-in does not wait for the 100 Continue response from the proxy server before sending the request content. You should use HTTP 1.1 100 Continue support when configuring the plug-in to work with certain types of proxy firewalls.

This property is ignored for POST requests to prevent a failure from occurring if the proxy server closes a connection because of a time-out.

This property is only valid for a proxy server, and applies to the Server element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is false.

com.ibm.ws.odr.plugincfg.property.ESIEnable

Use this property to enable or disable the Edge Side Include (ESI) processor. If the ESI processor is disabled, the other ESI elements in the file are ignored.

This property is only valid for a proxy server, and applies to the Property element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is true.

com.ibm.ws.odr.plugincfg.property.ESIMaxCacheSize

Use this property to specify, 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 to its expiration time.

This property is only valid for a proxy server, and applies to the Property element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is 1024.

com.ibm.ws.odr.plugincfg.property.ESIInvalidationMonitor

Use this property to specify whether or not the ESI processor receives invalidations from the proxy server.

This property is only valid for a proxy server, and applies to the Property element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is false.

com.ibm.ws.odr.plugincfg.property.https.keyring

Use this property to specify the directory location of the SAF keyring when the protocol of the transport is set to HTTPS.

This property is only valid for a proxy server, and applies to the Property element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `profileRoot/etc/plugin-key.kdb`.

com.ibm.ws.odr.plugincfg.property.https.stashfile

Use this property to specify the location of the stashfile.

This property is only valid for a proxy server, and applies to the Property element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is `profileRoot/node/etc/plugin-key.sth`.

com.ibm.ws.odr.plugincfg.property.PluginInstallRoot

Use this property to specify the installation path for the plug-in.

You must set this property, to the fully qualified path of the plug-in installation root. If you use the default value, the property does not display in the plugin-cfg.xml file.

This property is only valid for a proxy server, and applies to the Property element in the plugin-cfg.xml file that the proxy server automatically generates.

The default value is "".

com.ibm.ws.pm.checkingDBconnection

Use this property to specify whether the persistence manager is to continue checking the availability of a database, that was previously marked as unavailable, until a connection with that database is successfully established.

If a database service is down when the persistent manager attempts to establish a connection to that database, the database is marked as unavailable. Typically, the persistent manager does not re-attempt to establish a connection after a database is marked as unavailable. If you set this property to `true`, the persistence manager continues to check the availability of the database until it is able to successfully establish a connection to that database.

The default value for this property is `false`.

com.ibm.ws.runtime.component.ResourceMgr.postBindNotify

Use this property to make the Connection Factory MBeans available when a resource adapter starts. Typically, when a resource adapter starts, the Connection Factory MBeans are not available for the resource adapter to query. However, certain resource adapters, such as the IMS DB Resource Adapter, require this functionality for initialization.

If you are not using a resource adapter that requires the availability of Connection Factory MBeans at initialization, add this property to your JVM settings and set the value to `false`.

The default value for this property is `true`.

com.ibm.ws.runtime.logThreadPoolGrowth

Thread pools that are allowed to grow are configured with a maximum size but allowed to increase in size beyond that maximum. However, by default, no messages are issued that indicate that the maximum size has been exceeded.

Set this property to `true` if you want the server to send a message to the log file when a thread pool that is allowed to grow increases beyond its configured maximum size.

com.ibm.ws.scripting.apptimeout

Use this property to specify, in seconds, the length of time that can elapse before an application installation, or an application update times out. The default value is 86400, which is equivalent to 24 hours.

Specifying a reasonable value for this property prevents the installation, or update process from continuing indefinitely when a situation occurs that prevents the installation, or update script from completing. For example, you might have a JACL script that updates an EAR file that cannot complete because the deployment manager that the script is connected to stops.

com.ibm.ws.sib.webservices.useSOAPJMSTextMessages

By default on WebSphere Application Server Version 6 or later, a SOAP over JMS web service message sent by the web services gateway is sent as a `JmsBytesMessage`.

Set this property to `true` to modify the default behavior and send a compatible `JmsTextMessage`. Setting this property modifies the default behavior for all outbound JMS web services invocations sent from the service integration bus.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.use602RequiredAttrCompatibility

Use the `com.ibm.ws.use602RequiredAttrCompatibility` custom property to specify whether the `<required>` attribute is evaluated prior to other attributes in the `cachespec.xml` file.

In Version 6.0.2, if you set the `<required>` attribute to `false`, then all of the other attributes within the `cachespec.xml` file are ignored and a cache ID is generated.

Note: In later versions, by default, the `<required>` attribute is evaluated along with all of the other attributes to determine if a cache ID is generated.

Note: If you set the `com.ibm.ws.use602RequiredAttrCompatibility` custom property to `true`, then the behavior of the `cachespec.xml` file is reverted back to the behavior in Version 6.0.2. The `<required>` attribute is evaluated prior to other attributes in the `cachespec.xml` file. The default value for this custom property is `false`. When you set this JVM custom property, which only applies to the application server level, it affects all of the dynamic cache users.

If you decide to use this custom property, specify it as an application server JVM custom property unless otherwise indicated within the context of a specific task.

com.ibm.ws.webservices.allowNoSOAPActionHeader

Use this property to enable the web services engine to tolerate an incoming web service request that **does not** contain a SOAPAction header. This property must be set at the application server level.

The SOAP specification states that an HTTP request message must contain a SOAPAction HTTP header field with a quoted empty string value, if in the corresponding WSDL description, the `soapAction` of `soapbind:operation` is either not present, or present with an empty string as its value. However, if you want the web services engine to handle requests that do not contain a SOAPACTION header, add this property to the application server settings and set it to `true`.

When this property is not specified, or is not set to `true`, if an incoming SOAP request message does not contain a SOAPAction header, a SOAP Fault is returned to the client

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.webservices.allowStatusCode202OneWay

Use this property to allow a JAX-RPC one-way service to send a 202 status code instead of a 200 status code.

A JAX-RPC one-way service deployed on WebSphere Application Server normally returns a 200 HTTP status code. Some JAX-RPC implementations cannot tolerate a 200 status code, preferring a 202 instead. According to the Basic Profile Version 1.1, both 200 and 202 are valid status codes for one-way services.

If the property is set to true, then the JAX-RPC one-way service returns a 202 status code.

The default value is false.

This property only applies to the application server JVM.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.webservices.appendRootCauseToWSF

If you are a JAX-RPC user, use this property to loop through all the exception causes and concatenate the details into the Fault details in the response.

The Fault details in the response typically does not contain any information about the original exception. This lack of information can make problem determination more difficult if the developer does not have access to the logs from the service provider.

The default value is false.

This property only applies to the application server JVM.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.webservices.contentTransferEncoding

Use this property to specify a range of bits for which XML-encoding is disabled. Typically any integer that is greater than 127 is XML-encoded. When you specify this property:

- Web services disables encoding for integers that fall within the specified range.
- The HTTP transport message contains a ContentTransferEncoding header that is set to the value that is specified for this custom property.

Specify 7bit, if you only want integers greater than 127 encoded. Specify 8bit, if you only want integers greater than 255 encoded. Specify binary, if you want encoding disabled for all integers.

The default value is 7bit.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.webservices.disableSOAPElementLazyParse

Use this property to disable lazy parsing of SOAPElements. Lazy parsing is designed for situations where the client is not parsing the SOAPElement. If a client is parsing the SOAPElement with SAAJ, it is better to not delay parsing by the web services component.

You can set this property as a JVM custom property at either the server or client level. When this property is set at either the server or client level, the setting applies to all applications on the JVM. The default value for this property is `false`.

You can also use an application assembly tool to specify this property as a new web service description binding entry for the port component binding, if you want to disable lazy parsing of SOAP elements on an application-by-application basis for a particular server, instead of for all of the servers that are managed by the deployment manager.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

`com.ibm.ws.webservices.engine.transport.jms.propagateOneWaySystemExceptions`

Use this property to enable exceptions that occur during the processing of a one-way JMS Web service to be propagated back to the EJB container. This propagation makes normal error recovery possible.

If this property is set to `false`, an exception is wrapped in a `WebServicesFault` message and sent back to the client. Because the Web service is not aware of the exception, no recovery is attempted.

The default value for this property is `false`.

gotcha: This property does not apply to a one-way HTTP Web service, or to two-way JMS requests.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

`com.ibm.ws.webservices.HttpRedirectWithProxy`

Set this property to `true` to allow HTTP redirect requests to be sent through the proxy server. When you set this property to `true`, you change the default behavior for all outbound HTTP redirect requests sent from the JAX-RPC runtime. When this property is set to `false`, a redirect request is sent to a remote server directly even though a proxy server is configured.

The default value for this property is `false`.

If you decide to use this custom property, you must specify it as an proxy server JVM custom property.

`com.ibm.ws.webservices.ignoreUnknownElements`

Use this property to control whether clients can ignore extra XML elements that are sometimes found within literal SOAP operation responses.

Setting this property to `true` provides you with the flexibility of being able to update your server code to include additional response information, without having to immediately update your client code to process this additional information. However, when this functionality is enabled, the checking of SOAP message against the expected message structure is more relaxed than when this property is set to `false`.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

`com.ibm.ws.webservices.jaxrpc.parse.tolerate.invalid.namespace`

Use this property to enable the JAX-RPC engine to use a more tolerant algorithm when determining whether to accept an incoming JAX-RPC message.

Typically, if an incoming JAX-RPC message uses an invalid namespace for a body element, the JAX-RPC engine rejects the message. If you set this property to `true`, the JAX-RPC engine uses a more tolerant algorithm that ignores the namespace mismatch.

The default value is false

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.webservices.resolveXMLSchemaDTD

Use this custom property to enable a JAX-RPC application to properly start even if the schema or WSDL file that is represented in the `_AbsoluteImportResolver` class also references the `http://www.w3.org/2001/XMLSchema.dtd` DTD.

When you run on a host that is not connected to the Internet, a JAX-RPC application that is packaged with the `_AbsoluteImportResolver` class might not start properly. The following error might exist in the log files:

```
WSDDPort      W com.ibm.ws.webservices.engine.deployment.wsdd.WSDDPort expand
WSWS3114E: Error: Internal error.
java.net.UnknownHostException: www.w3.org
```

Setting this custom property to true enables a JAX-RPC application to properly start even if the schema or WSDL file that is represented in the `_AbsoluteImportResolver` class also references the `http://www.w3.org/2001/XMLSchema.dtd` DTD.

The default value is false

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.webservices.searchForAppServer

Use this property to control whether `DualMetaDataLoaderImpl E loadWebContainerPorts` could not find any http or https ports messages are sent to the system log.

Depending on your system configuration, if a web services application is installed across both a web server and an application server, your system might issue this message, indicating that an error occurred even though this is a valid configuration. Therefore if you install any of your web services applications across both a web server and an application server, you might not want these messages sent to the system log.

If the `com.ibm.ws.webservices.searchForAppServer` property is set to true, any `DualMetaDataLoaderImpl E loadWebContainerPorts` could not find any http or https ports messages that are issued are not sent to the system log. If this property is not specified or is set to false, these messages are sent to the system log.

The default value for this property is false.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.webservices.serialize.2DimArray.asArrays

Use this property to cause the JAX-RPC runtime to serialize two-dimensional XML arrays as a series of arrays.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

The default value for this property is false.

The following message snippet illustrates a series of elements, which is a valid format for representing two-dimensional XML arrays when this property is set to false.

```

<p565:sayHelloResponse xmlns:p565="http://ibm.com">
  <sayHelloReturn xsi:type="soapenc:Array"
soapenc:arrayType="xsd:string[2,3]">
    <item xsi:type="xsd:string">array1 element1</item>
    <item xsi:type="xsd:string">array1 element2</item>
    <item xsi:type="xsd:string">array1 element3</item>
    <item xsi:type="xsd:string">array2 element1</item>
    <item xsi:type="xsd:string">array2 element2</item>
    <item xsi:type="xsd:string">array2 element3</item>
  </sayHelloReturn>
</p565:sayHelloResponse>

```

The following message snippet illustrates an array of two arrays, with each array containing three elements, which is a valid format for representing two-dimensional XML arrays when this property is set to true.

```

<p565:sayHelloResponse xmlns:p565="http://ibm.com">
  <sayHelloReturn xsi:type="soapenc:Array"
soapenc:arrayType="xsd:string[][2]">
    <item soapenc:arrayType="xsd:string[3]">
      <item>array1 element1</item>
      <item>array1 element2</item>
      <item>array1 element3</item>
    </item>
    <item soapenc:arrayType="xsd:string[3]">
      <item>array2 element1</item>
      <item>array2 element2</item>
      <item>array2 element3</item>
    </item>
  </sayHelloReturn>
</p565:sayHelloResponse>

```

com.ibm.ws.webservices.serializeDetailElementUsingDefaultNamespace

Use this property to specify whether the application server uses an actual prefix name to locate the namespace that defines the Fault detail, or uses a default namespace to define the Fault detail.

When a JAX-RPC Web service responds with a SOAP Fault, an actual prefix name is typically used to locate the namespace that defines the contents of the Fault detail. Following is an example of the message that the application server typically issues in this situation:

```

<soapenv:Fault
  xmlns:soapenv=
    "http://schemas.xmlsoap.org/soap/envelope/">
  <faultcode xmlns="http://sample">
    sampleFault
  </faultcode>
  <faultstring>sample text</faultstring>
  <detail encodingStyle="">
    <sampleFault
      xmlns="http://sample">
      ...
    </sampleFault>
  </detail>
</soapenv:Fault>

```

If your application server needs to communicate with .Net clients, and these .Net clients require the use of a default namespace to define the contents of the Fault detail, set this property to true. When this property is set to true, the message that the application server issues is similar to the message that was sent from a version previous to a Version 6.x application server.

The default value for this property is false.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.webservices.suppressHTTPRequestPortSuffix

Use this property to control whether a port number can be left in an HTTP POST request that sends a SOAP message.

Some web service implementations do not properly tolerate the presence of a port number within the HTTP POST request that sends the SOAP message. If you have a web service client that needs to inter-operate with web service that cannot tolerate a port number within an HTTP POST request that sends a SOAP message, set this custom property to `true`.

When you set this property to `true`, the port number is removed from the HTTP POST request before it is sent.

gotcha: You must restart the server before this configuration setting takes affect.

The default value for this custom property is `false`.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.websvcs.attachments.sizethreshold

Use this property to specify, in bytes, the maximum size of an attachment on the JAX-WS client or service that can be written to memory. By default, the maximum attachment size is set to 102400 bytes. With this value, if an attachment exceeds 100 KBs, it is cached to the file system instead of written to memory. When you use this custom property, as you increase the maximum cache size, there is a greater impact on performance and, potentially, to the Java heap.

Note: To specify the maximum size of an attachment on the JAX-RPC client or service, see the `com.ibm.websphere.webservices.attachments.maxMemCacheSize` custom property.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.websvcs.suppressHTTPRequestPortSuffix

Use this property to prevent the JAX-WS runtime from appending the port number to the HTTP Host header value to a request.

A JAX-WS client might receive a `java.io.IOException` in response to a request, especially if there is a non-IBM web server located between the client and the web service the client is trying to call. This intermediary server might not understand where to route the request because the JAX-WS runtime has appended the port number to the HTTP Host header value. For example, JAX-WS runtime might have changed the header value from the endpoint URL `lilygirl.austin.mycompany.com` to the URL `lilygirl.austin.mycompany.com:80`, which includes the port number.

To prevent the JAX-WS runtime from appending the port number to the HTTP Host header value, add this custom property to your JVM settings, and set it to `true`. When this property is set to `true`, the Host header only contains the hostname of the endpoint URL; for example, `Host: lilygirl.austin.mycompany.com`

The default value for this property is `false`, which means that, the port number is appended to a Host header value.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.websvcs.unmanaged.client.dontUseOverriddenEndpointUri

You might want a request by an unmanaged JAX-WS client service to be sent to the endpoint URL that is specified in the **Overridden endpoint URL** field on the administrative console. The value of this managed

field, which is set as part of the web services client port configuration, overwrites the endpoint that is specified in the WSDL file. For more information on this field, read about the web services client port.

Note: If you have either all managed clients or a mixture of both managed and unmanaged clients, you can edit the **Overridden endpoint URL** field in the administrative console. However, if you do not have any managed clients, you cannot edit the field.

Normally, you do not want an unmanaged JAX-WS client service to access this managed client service function. However, you might depend on unmanaged JAX-WS client services accessing this URL. By default, the `com.ibm.ws.websvcs.unmanaged.client.dontUseOverriddenEndpointUri` custom property is set to `false` to allow unmanaged JAX-WS client services to access the endpoint URL that overwrites the endpoint in the WSDL file.

This custom property is set on the application server level where a JAX-WS client is installed or a Java EE client exists if you run the `launchClient`.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.ws.ws.wsba.protocolmessages.twoway

Use this property to improve the performance of an application server that is handling requests for Web Services Business Activities (WS-BA). Specifying `true` for this custom property improves application server performance when WS-BA protocol messages are sent between two application servers. The default value for this property is `true`.

gotcha: If you decide to use this custom property, the property must be set on the application server that initiates the requests. It does not have to be set on the application server that receives the requests.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

com.ibm.wsspi.amm.merge.ignoreValidationExceptions

Use this custom property to indicate to the JVM that it should ignore validation exceptions that might occur during EJB processing. When an application is configured with necessary classes defined in shared libraries during EJB processing, incomplete information may be generated. As a result, a validation exception might occur and the following exception message may appear:

```
AnnotativeMetadataManagerImpl merge caught exception while merging com.ibm.wsspi.amm.validate.ValidationException:
the interface com.xyz.app.myappRemote does not define a
valid remote business interface; the method mygetMethod does not
conform to RMI rules.
```

Set this property to `true` if you want the JVM to ignore these validation exceptions.

The default value is `false`.

If you decide to use this custom property, you must specify it as a JVM custom property for the application server.

com.ibm.xml.xlpx.jaxb.opti.level

Use the `com.ibm.xml.xlpx.jaxb.opti.level` custom property to control whether optimization methods are enabled for Java Architecture for XML Binding (JAXB) unmarshalling (deserialization) and marshalling (serialization). The following table lists the supported values for this custom property and their effect on applications and web services.

Table 32. Supported values for the custom property. The table includes the custom property value and the effect of the custom property on applications and web services.

Custom property value	Effect
com.ibm.xml.xlsp.jaxb.opti.level=0	Optimization methods are not enabled.
com.ibm.xml.xlsp.jaxb.opti.level=1	Only unmarshalling optimization methods are enabled.
com.ibm.xml.xlsp.jaxb.opti.level=2	Only marshalling optimization methods are enabled.
com.ibm.xml.xlsp.jaxb.opti.level=3	Both unmarshalling and marshalling optimization methods are enabled, which is the default value.

For optimum performance, set the custom property value to 3. This value increases throughput for web services and applications that use JAXB directly. If you are experiencing issues with optimization after setting this value, change the value to 0 as a temporary workaround.

You can set this custom property on the application server level only.

config_consistency_check

Use this property to optionally turn off the default workspace consistency process. The deployment manager maintains a master configuration repository for the entire cell. By default, when the configuration changes, the product compares the configuration in the workspace with the master repository to maintain workspace consistency. However, the consistency verification process can cause an increase in the amount of time 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`.

Note: The `config_consistency_check` custom property affects the deployment manager process only. It does not affect other processes including the node agent and application server processes. The consistency check is not performed on these processes. However, within the `SystemOut.log` files for these processes, you might see a note that the consistency check is disabled. For these non-deployment manager processes, you can ignore this message.

deactivateWildcardURIMapping

Use this property to enable the `plugin-cfg.xml` file generator to recognize the URI patterns specified on the `file.serving.patterns.allow` attribute in the `ibm-web-ext.xmi` file for a web application.

The `plugin-cfg.xml` file generator only recognizes the URI patterns specified on the `file.serving.patterns.allow` attribute if the `FileServingEnabled` attribute in that `ibm-web-ext.xmi` file is set to `true`. However, when the `FileServingEnabled` attribute is set to `true`, the `plugin-cfg.xml` file generator automatically adds the wildcard URI mapping, `/*`, to the `plugin-cfg.xml` file, which negates the usefulness of defining unique file serving patterns.

Setting the `deactivateWildcardURIMapping` property to `true` prevents the `plugin-cfg.xml` file generator from adding the `/*` to the `plugin-cfg.xml` file, and enables the `plugin-cfg.xml` file generator to recognize the URI patterns specified on the `file.serving.patterns.allow` attribute. If this property is not added to the JVM settings, or is set to `false`, the `/*` is automatically added to the `plugin-cfg.xml` file.

disableWSAddressCaching

Use this property to disable address caching for web services. If your system typically runs with lots of client threads, and you encounter lock contention on the wsAddrCache cache, you can set this custom property to true, to prevent caching of the web services data.

The default value for this property is false.

DRS_THREADPOOL_MINSIZE

Specifies the minimum number of threads to allow in the data replication service (DRS) thread pool.

When an application server starts, threads are not initially assigned to the thread pool. Threads are added to the thread pool as the workload that is assigned to the application server requires them and until the number of threads in the pool equals the number of threads that are specified by this custom property. After this point in time, additional threads are added and removed as the workload changes. However, the number of threads in the pool never decreases below the number that is specified by this custom property, even if some of the threads are idle.

The default value for this custom property is 40 threads.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

DRS_THREADPOOL_MAXSIZE

Specifies the maximum number of threads to maintain in the DRS thread pool.

The default value for this custom property is 100 threads.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

DRS_THREADPOOL_ISGROWABLE

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

The maximum number of threads that can be created is constrained only within the limits of the Java virtual machine and the operating system. When a thread pool, that is allowed to grow, expands beyond the maximum size, the additional threads are not reused and are discarded from the pool after processing the work items for which they were created is completed.

The default value for this custom property is false.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

java.util.logging.configureByLoggingPropertiesFile

Use this custom property to specify whether the JVM uses the logging.properties file to configure JSR-47 logging.

If this property is not added to the JVM configuration settings, or is set to false, the configuration settings contained in the logging.properties file are not picked up because the product overrides the base JSR47 logging configuration with the java.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager system property setting. In this situation, only logging settings that can be changed programmatically, such as the addition of handlers, and formatters, can be modified.

When this property is set to true, the JVM still configures the WsLogManager as the LogManager, but during server startup, the logging configuration for applications using JSR-47 logging is initialized based on

settings in the logging.properties file. Refer to the Java Utility Logging API documentation for valid logging properties and format that can be specified in the logging.properties configuration file.

gotcha: Do not assign java.util.logging.ConsoleHandler to any of the loggers because this assignment can cause an infinite loop as mentioned in the Java logging topic.

The logging.properties file is located in the <<WAS_install>>/java/jre/lib/logging.properties directory, and can be customized as needed.

The default setting for this property is false.

jaxws.ignore.extraWSDLOps

Use this property if there are more operations in the WSDL than built into the client.

Default client behavior is to validate the operations built into the client against the operations in WSDL and fail if they do not match. Set this property to true if there are more operations in the WSDL than built into the client and the WSDL validation will succeed and the client can be invoked.

The default value is false.

jaxws.payload.highFidelity

Use this property to enable lossless transformations. When this property is set to true, the Web Service runtime guarantees that the incoming message and the message at the SOAPHandler boundary are the same.

Typically, the SOAP message received by a JAX-WS SOAPHandler is not exactly the same as the inbound SOAP message. For example, the message received by the JAX-WS SOAPHandler might use different xml prefixes than the original inbound message. These subtle changes do not affect the logical interpretation of the message. However, you must add this property to your JVM settings and set the property to true if messages at the SOAPHandler boundary must be exactly the same as the incoming messages. For example, the Canonicalization Specification (C14N) requires that the prefix names are preserved.

bprac: You should only use this property if your SOAP requests access the contents of a soapenv:Body element within your SOAPHandlers. Setting the property to true might degrade Web Service runtime performance.

The default value for this property is false.

jaxws.provider.interpretNullAsOneway

If you have a JAX-WS web service that defines a Provider-based endpoint using the javax.xml.ws.Provider annotation and a WSDL file is not specified, you can use this custom property to control how the JAX-WS runtime environment behaves when the Provider returns a null value from the invoke() method. By default, the runtime environment will send back a response that consists of a SOAPEnvelope that contains an empty SOAPBody element.

If this property is set to true, whenever the Provider implementation returns a null value and a WSDL file is not defined, the runtime environment interprets the null value returned from the Provider implementation as a request-only operation so that no response is returned. As with all request-only operations, some qualities of services, such as WS-Transactions, will not be available.

If the javax.xml.ws.WebServiceProvider annotation specifies a WSDL value and the WSDL defines a request and response operation, the JAX-WS runtime environment always returns a response that consists of a SOAPEnvelope that contains an empty SOAPBody, regardless of the setting of this property.

The default value for this custom property is false.

You must set this property to true for each server that requires a change in the default value.

jaxws.runtime.restrictStaticWebmethod

Use this property to prevent exposure of static operations. When this property is set to true, the JAX-WS runtime prevents the exposure of static operations.

The default value is false.

jaxws.soapfault.local.exceptions.disable

Note: Use the `jaxws.soapfault.local.exceptions.disable` property to prevent locally occurring exceptions on a JAX-WS client from being treated as a `SOAPFault`. By default, if a JAX-WS client encounters a local exception, a `SOAPFault` is created for the exception. An example of a local exception is a `ConnectException` caused by an invalid host or port. The relevant JAX-WS application handlers `handleFault` methods are called with the `SOAPFault`, then a `SOAPFaultException` is thrown back through the JAX-WS client's invoked method.

By setting this property to true, local exceptions create an empty message. The relevant JAX-WS application handlers `handleMessage` methods are called with the empty message, then a `WebServiceException` is thrown back through the JAX-WS client's invoked method. This was the behavior in previous releases.

The default value for this property is false.

ODCClearMessageAge

Use this property to establish a length of time, specified in milliseconds, after which an ODC message is removed from the bulletin board, even if the receiver has not acknowledged the message. Specifying a value for this property helps prevent the build up of messages that, for some reason, do not get acknowledged.

You can specify any positive integer as a value for this property, but a value of 300000 (5 minutes) or higher is recommended to avoid premature removal of messages.

The default value is 300000 milliseconds.

ODCInit.disabled

Set this property to true if you want to disable the communication between processes for the On Demand Configuration (ODC) component, and for all local ODC processing.

The on demand configuration component is used when deploying Web services-based applications, and when using a WebSphere Application Server Proxy Server to handle requests. The on demand configuration component is enabled or disabled on a cell-wide basis. Therefore, if your topology contains any proxy servers, or any web services based applications, you should not disable the on demand configuration service.

If you are running in a large topology environment where Web services-based applications are not deployed, or WebSphere Application Server Proxy Servers are not used to handle requests, the on demand configuration component is not utilized, and you can set this property to true. Setting this property to true disables the on demand configuration component, which will reduce network bandwidth and CPU utilization.

The default value is false.

org.eclipse.jst.j2ee.commonarchivecore.disableZip

Use this custom property to allow ZIP archives to be processed as simple files.

Set this property to true to allow ZIP archives to be processed as simple files when scanning the files of a deployed application.

The default value is false.

This property must be set as a custom property for the IBM WebSphere Application Server process which runs applications for which ZIP files are to be ignored.

org.eclipse.jst.j2ee.commonarchivecore.FILTERBINARIES

Use this custom property to allow certain application files from being listed during runtime processing.

Because of new JavaEE5 annotations processing requirements, more application files are typically listed during runtime processing than are listed in previous versions of the product. The additional listing might cause applications that are migrated from previous versions of the product to start more slowly, as additional time is spent listing application files.

The default value is not set.

If you decide to use this custom property, you must specify it as an application server JVM custom property.

sizeThreshold

Use this property when you want to control the algorithm for caching attachments in the JAX-WS runtime environment. When SOAP messages are processed by the JAX-WS runtime environment, the runtime environment stores small attachments in memory and stores large attachments in a file on a disk.

Use this property to specify, in kilobytes, the maximum size of an attachment that can be written to memory. If you do not specify a value for this property, the default value is 32. This default value specifies that any attachment that is less than 32 KB is stored in memory.

If the value of this property is increased, larger attachments are stored in memory. Increased values might increase the performance of the web service; however, this increased value causes the Java heap to grow. Setting the value too high might cause `OutOfMemoryError` errors to occur.

When determining a value for this property, remember that the larger the maximum cache size, the greater the impact on performance, and, potentially, to the Java heap. Use this property only for Java and performance tuning.

This property does not affect the logical processing of JAX-WS web services. Your JAX-WS web services will successfully process SOAP messages containing both large and small attachments, regardless of the setting of this property.

threadpool.maxsize

The JVM custom property, **threadpool.maxsize**, is a dedicated string used to start each application server. **threadpool.maxsize** controls the number of application servers that are started in parallel. You add **threadpool.maxsize** on the Node Agent to provide thread pool size as follows:

System Administration > Node agent > *nodeagent_name* > Configuration > Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties > New. Then you supply the name, **threadpool.maxsize** and a value.

If the value you supply is greater than 0, then a threadpool is created with that value as its maximum value. For example, if the value you supply is 3, a maximum of 3 application servers will be started in parallel. If the value you supply is 0 or less than 0, then the Node Agent behaves by launching application servers sequentially.

There is no default value. If you choose to use **threadpool.maxsize**, you must supply a value.

Note: When using the administrative console, you cannot leave the **threadpool.maxsize** value field blank. If you pass a blank value to the property from the command line then it will be considered as an illegal value, and the default behavior of the Node Agent is restored.

There is no maximum value for **threadpool.maxsize**. The maximum threads created will be equal to the number Application Servers. You cannot configure this value.

webservices.unify.faults

Note: Use the `webservices.unify.faults` property to disable SOAP Fault unification for JAX-WS and JAX-RPC. By default, the web service runtime environments (both JAX-WS and JAX-RPC) unify all faults generated by the runtime environment to a single type of fault containing a `faultcode` of `Server` and a `faultstring` of `Internal Error`. The faults do not contain any additional information identifying the actual cause of the fault. The unification of faults results in a more secure system, preventing detailed information regarding why inbound message processing failed from being returned to message senders.

The default value for this property is `true`, which causes faults to be unified. If your applications require fault details, then you can set this property to `false` to disable fault unification, allowing detailed information to be returned in faults. Note that regardless of the property setting, checked exceptions defined in the WSDL and thrown by a service provider method implementation are not unified. Additionally, detailed information regarding the cause of the fault are logged if trace is enabled, regardless of the setting of this property.

This property and the associated behavior is new in Version 8 of the product.

wink.client.readTimeout

Use this property to specify how long the `RestClient` object waits (in milliseconds) for a response to requests before timing out. A value of zero (0) means that the client waits for an unlimited amount of time and will not timeout.

The default value is 60,000 milliseconds.

wink.client.connectTimeout

Use this property to specify how long the `RestClient` object waits (in milliseconds) before timing out when attempting to connect to the target resource. A value of zero (0) means that the client waits for an unlimited amount of time and will not timeout.

The default value is 60,000 milliseconds.

Tuning application servers

The product contains interrelated components that must be harmoniously tuned to support the custom needs of your end-to-end e-business application.

About this task

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.

Procedure

1. Run the `applyPerfTuningTemplate.py`, as the starting point for improving the performance of an application server.

You can use the python-based tuning script, `applyPerfTuningTemplate.py`, along with one of its template files, to apply recommended performance tuning settings. The script, and these template files are located in the `<WAS_HOME>/scriptLibraries/perfTuning/V70` directory.

2. **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 the topic about 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 information about Object Request Broker custom properties. *Administering applications and their environment* PDF.

3. **Tune the XML parser definitions.**

- **Description:** Facilitates server startup by adding XML parser definitions to the `jaxp.properties` and `xerces.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.DocumentBuilderFactory=org.apache.xerces.jaxp.
    DocumentBuilderFactoryImpl
org.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.XIncludeAwareParserConfiguration
```

You can also consult with the `jre/lib/jaxp.properties` and `jre/lib/xerces.properties` files that come with the JDK installation. These sample files always contain the recommended settings.

- **Default value:** None
- **Recommended value:** None

4. **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.

5. **Tune the web container.** The product 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 the product 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 provides the best throughput. The number of threads configured does not represent the number of requests that the product can process concurrently. Requests are queued in the transport chain when all threads are busy. To specify the thread pool settings:
 - a. Click **Servers > Server Types > WebSphere application servers > server_name Web container settings > Web container > Web container transport chains**.
 - b. Select the normal inbound chain for serving requests. This chain is typically called `WCInboundDefault`, and listens 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 the product closes a given client connection after a number of requests or a timeout period. After a connection is closed, it is 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 > Server Types > WebSphere application servers >server_name**. Then in the Container Settings section, click **Web container > Web container transport chains**.
 - b. Select the normal inbound chain for serving requests. This chain is typically called WCInboundDefault, and listens on port 9080.
 - c. Click **HTTP Inbound Channel (HTTP_2)**.
 - d. Enter values for **Maximum persistent requests** and **Persistent timeout**.
- 6. **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.

7. **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.

8. **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.
9. **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.

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.

Tuning the application server using pre-defined tuning templates

You can use the python-based tuning script, `applyPerfTuningTemplate.py`, along with one of its template files, to apply pre-defined performance tuning templates to your application server or cluster. The script, and these property-based template files are located in the `<WAS_HOME>/scriptLibraries/perfTuning/V70` directory.

Before you begin

bprac: The configuration settings applied by this script and the associated tuning templates should be viewed as potential performance tuning options for you to explore or use as a starting point for additional tuning. The configuration settings that each of the pre-defined templates applies are geared towards optimizing common application server environments or scenarios. Typically, these settings improve performance for many applications.

Because optimizing for performance often involves trade-offs with features, capabilities, or functional behavior, some of these settings might impact application correctness, while other

settings might be inappropriate for your environment. Please review the documentation below and consider the impact of these settings to your application inventory and infrastructure.

As with any performance tuning exercise, the settings configured by the predefined templates should be evaluated in a controlled preproduction test environment. You can then create a customized template to refine the tuning settings to meet the specific needs of your applications and production environment.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log , SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Typically, when you run the applyPerfTuningTemplate.py script, you will specify either the production.props template file or the development.props template file to apply against the target server or cluster.

- If you specify the production.props template file when you run the applyPerfTuningTemplate.py script, the script applies configuration settings that are appropriate for a production environment where application changes are rare and optimal runtime performance is important.
- If you specify the development.props template file when you run the applyPerfTuningTemplate.py script, the script applies configuration settings that are appropriate for a development environment where frequent application updates are performed and system resources are at a minimum.

In addition to these two common templates, a third template file, default.props, is provided to enable you to revert the server configuration settings back to the out-of-the-box defaults settings.

You can also create your own custom tuning template. To create a custom tuning template, copy one of the existing templates, modify the configuration settings to better fit the needs of your applications and environment, and then use the applyPerfTuningTemplate.py script to apply these customized settings. The script and properties files leverage the property file configuration management features that wsadmin provides, and can easily be augmented to tune additional server components. See the topic Using properties files to manage system configuration for more information.

About this task

Review the following table to see the configuration changes that occur based on the template file that you specify when you run the applyPerfTuningTemplate.py script. A blank cell in this table indicates that the listed parameter is not configured, or is configured back to the default settings for the server defaults.

Table 33. Tuning parameters and their template values. The table includes the tuning parameter and its value for the default template, the production template and the development template.

Parameter	Server default (default.props template file)	Production environment (production.props template file)	Development environment (development.props template file)
JVM Heap Size (MB)	50 min / 256 max	512 min / 512 max	256 min / 512 max
See the topic Tuning the IBM virtual machine for Java for more information about this setting.			

Table 33. Tuning parameters and their template values (continued). The table includes the tuning parameter and its value for the default template, the production template and the development template.

Parameter	Server default (default.props template file)	Production environment (production.props template file)	Development environment (development.props template file)
Verbose GC See the topic Tuning the IBM virtual machine for Java for more information about this setting.	disabled	enabled	enabled
JVM Diagnostic Trace (Generic JVM Arguments) See the topic Tuning the IBM virtual machine for Java for more information about this setting.		-Dcom.ibm.xml.xlsp.jaxb .opti.level=3	-Dcom.ibm.xml.xlsp.jaxb .opti.level=3
HTTP (9080) and HTTPS (9443) Channel maxKeepAliveRequests See the topic HTTP transport custom properties for more information about this setting.	100	10000	10000
TCP Channel maxOpenConnections	20000	500	500
TCP Channel listenBacklog	511	128	128
Development Mode See the topic Application server settings for more information about this setting.	disabled		enabled
Server Component Provisioning See the topic Application server settings for more information about this setting.	disabled	enabled	enabled
PMI Statistic Set See the topic Enabling PMI data collection for more information about this setting.	basic	none	none
Authentication Cache Timeout See the topic Authentication cache settings for more information about this setting.	10 minutes	60 minutes	60 minutes

Table 33. Tuning parameters and their template values (continued). The table includes the tuning parameter and its value for the default template, the production template and the development template.

Parameter	Server default (default.props template file)	Production environment (production.props template file)	Development environment (development.props template file)
Data Source Connection Pool Size* See the topic Connection pool settings for more information about this setting.	1 min / 10 max	10 min / 50 max	
Data Source Prepared Statement Cache Size* See the topic WebSphere Application Server data source properties for more information about this setting.	10	50	
ORB Pass-by-Reference** See the topic Request Broker service settings for more information about this setting.	disabled	enabled	enabled
Web Server Plug-in ServerIOTimeout	900	900	900
Thread Pools (Web Container, ORB, Default) See the topic Thread pool settings for more information about this setting.	50 min / 50 max, 10 min / 50 max, 20 min / 20 max		5 min / 10 max

Table notes:

* Indicates items that are tuned only if they exist in the configuration. For example, a data source connection pool typically does not exist until an application is installed on the application server. If these items are created after your run the script, they are given the standard server default values unless you specify other settings.

** Enabling ORB Pass-By-Reference can cause incorrect application behavior in some cases, because the Java EE standard assumes pass-by-value semantics. However, enabling this option can improve performance up to 50% or more if the EJB client and server are installed in the same instance, and your application is written to take advantage of these feature. The topic Object Request Broker service settings can help you determine if this setting is appropriate for your environment.

Following are a few subtle platform-specific tuning differences:

Procedure

- Start the wsadmin tool if it is not already running, and then complete one of the following actions to tune an application server or all of the application servers in a cluster.
- Run the applyPerfTuningTemplate.py script to tune a specific server or cluster of servers running in a production environment.

```
wsadmin -f applyPerfTuningTemplate.py
[-nodeName node_name -serverName server_name][clusterName cluster_name] -templateFile production.props
```

- Run the `applyPerfTuningTemplate.py` script to tune a specific server or cluster of servers running in a development environment.

```
wsadmin -f applyPerfTuningTemplate.py  
[-nodeName node_name -serverName server_name][clusterName cluster_name] -templateFile development.props
```

- Run the `applyPerfTuningTemplate.py` script to change the settings for a server or a cluster back to the standard out-of-the-box default configuration settings.

```
wsadmin -f applyPerfTuningTemplate.py  
[-nodeName node_name -serverName server_name][clusterName cluster_name] -templateFile default.props
```

What to do next

Conduct a performance evaluation, and tuning exercise to determine if you should further fine tune the server for your specific applications.

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 enabling 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 enable 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 `localLogFilenamePrefix` 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 localLogFilenamePrefix 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.

Important: If you specify a value for the localLogFilenamePrefix custom property, you must also set the accessLogFileName HTTP channel custom property to the fully qualified name of the log file you want to use for in process requests. You cannot specify a variable, such as \$(SERVER_LOG_ROOT), as the value for this custom property.

Appendix. Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

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 products or components require multiple locations.

app_client_root

The default installation root directory for the Application Client for IBM WebSphere Application Server is the /QIBM/ProdData/WebSphere/AppClient/V8/client directory.

app_client_user_data_root

The default Application Client for IBM WebSphere Application Server user data root is the /QIBM/UserData/WebSphere/AppClient/V8/client directory.

app_client_profile_root

The default Application Client for IBM WebSphere Application Server profile root is the /QIBM/UserData/WebSphere/AppClient/V8/client/profiles/*profile_name* directory.

app_server_root

The default installation root directory for WebSphere Application Server is the /QIBM/ProdData/WebSphere/AppServer/V8/Base directory.

java_home

Table 34. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32-bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit
64-bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit

plugins_profile_root

The default Web Server Plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/*profile_name* directory.

plugins_root

The default installation root directory for Web Server Plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V8/webserver directory.

plugins_user_data_root

The default Web Server Plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver directory.

product_library

product_lib

This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS8x (where x is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is QWAS8A. The *app_server_root*/properties/product.properties file contains the value for the product library of the installation, was.install.library, and is located under the *app_server_root* directory.

profile_root

The default directory for a profile named *profile_name* for WebSphere Application Server is the `/QIBM/UserData/WebSphere/AppServer/V8/Base/profiles/profile_name` directory.

shared_product_library

The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root

The default user data directory for WebSphere Application Server is the `/QIBM/UserData/WebSphere/AppServer/V8/Base` directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root

The default web server path is `/www/web_server_name`.

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.

APACHE INFORMATION. This information may include all or portions of information which IBM obtained under the terms and conditions of the Apache License Version 2.0, January 2004. The information may also consist of voluntary contributions made by many individuals to the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org>. You may obtain a copy of the Apache License at <http://www.apache.org/licenses/LICENSE-2.0>.

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

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. For a current list of IBM trademarks, visit the IBM Copyright and trademark information Web site (www.ibm.com/legal/copytrade.shtml).

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

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Index

A

- administering
 - nodes 25
 - host names 25
 - stand-alone nodes 27
 - resources 25
 - topology 66
- administrative agents 28
 - administering
 - nodes 27
 - environment setup 30
 - node collection 36
 - security 29
 - settings 35
 - starting 33
 - stopping 33
 - unregistered nodes 36
- administrative services
 - configuration 38
 - custom properties 65
 - settings 64
- application servers
 - administering 81
 - configuration 149
 - UCS Transformation Format 151
 - generic servers 157
 - starting 159
 - stopping 159
 - management 109
 - ports 114, 146
 - processes
 - process definition 194
 - requests
 - stops processing 147
 - restarting
 - in recovery mode 124
 - running user profiles 125
 - settings 112
 - custom properties 117
 - custom services 193
 - DCS transport channels 178
 - environment entries 121
 - generic servers 159
 - host aliases 87
 - HTTP transport channels 170
 - HTTP transports 164
 - MIME types 89
 - ports 115
 - process execution 197
 - process logs 198
 - server components 118
 - TCP transport channels 175
 - thread pools 119
 - starting 122
 - stopping 126
 - tuning 245

C

- configuration
 - administrative services 38

D

- directory
 - installation
 - conventions 34, 123, 152, 156, 199, 253

E

- extension MBean providers 49, 62
 - settings 50, 62
- extension MBeans 50, 63
 - settings 50, 63

F

- file synchronization
 - remote services 38
- file transfer
 - remote services 38

H

- host aliases 87
- HTTP transport channels
 - custom properties 183

I

- IBM Toolbox
 - JDBC 95
- Inter-Process Communications
 - connectors
 - properties 48, 61

J

- JMX
 - connectors 47, 59
 - properties 40, 52
 - settings 48, 60
- JVM
 - configuration 201
 - custom properties 207
 - sendRedirect calls 206
 - settings 201

M

- messages
 - administrative audit 51, 63
- MIME types 88

N

- nodes
 - administering 25
 - administrative agents 27
 - administrative agents 36
 - host names 25
 - managed servers 25

P

- properties
 - user.timezone 130

Q

- QWAS8 system 153

R

- registered nodes
 - settings 36
- repository services
 - settings 39
- resources for learning
 - administrative topology 66
- runtime components
 - problems 126

S

- security
 - administrative agents 29
- servers 110
- services
 - custom properties 191
- shared library reference and mapping
 - settings 104
- SOAP
 - connectors
 - properties 48, 61

- starting administrative agents 33
- stopping administrative agents 33

T

- TCP transport channels
 - custom properties 186
- thread pools
 - settings 119
- time zones
 - application servers
 - user profiles 128
 - JVM processes 129
 - properties 130
- transport chains 162
 - configuration 160
 - disabling ports 190
- transport channels
 - HTTP channels 170
- tunnel transports
 - HTTP channels 170

U

- unregistered nodes
 - administrative agents 36

V

- virtual hosts 83

W

- web modules
 - requests
 - stop processing 147
- WebSphere Application Server
 - variables 89, 91, 92
- WebSphere variables 89