IBM Systems - iSeries

e-business and Web serving WebSphere Application Server - Express Version 5 Administration

Version 5 Release 4

IBM Systems - iSeries

e-business and Web serving
WebSphere Application Server - Express Version 5
Administration

Version 5 Release 4

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 159.

Third Edition (February 2006)

This edition applies to version 5.0 of IBM WebSphere Application Server - Express for iSeries (product number 5722-IWE) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Administration

The administrative interfaces for WebSphere(R) Application Server - Express allow you to configure and manage your application server, deploy and run applications, and configure security settings for your application server. For information on using the administrative tools, see "Administrative tools" on page 82.

WebSphere Application Server - Express includes these administrative tools:

- The HTTP Server Administration interface
- The WebSphere administrative console
- Qshell scripts
- The wsadmin administrative tool

Some iSeries(TM) servers are preinstalled with WebSphere Application Server - Express. To find out if you have a preinstalled system, see Determine whether or not WebSphere Application Server - Express is preinstalled on your iSeries server. If the product is preinstalled, see this topic for additional information:

"Administration considerations for preinstalled systems" on page 2
If you have a system on which WebSphere Application Server - Express is preinstalled, see this topic for information on checking the product configuration.

These topics describe how to perform the administrative tasks for WebSphere Application Server - Express.

"Create a new application server" on page 2
This topic describes how to use the HTTP Server Administration interface to create an application server. It also provides information on deleting an application server instance.

"Start and test your application server" on page 8
This topic describes how to start your application server and use the ExpressSamples sample application to verify that the application server is running correctly. It also provides information on stopping and restarting application servers.

"Configure the application server to run your application" on page 15
This topic describes how to configure the application server and application server resources for your application. It also provides information on enabling and disabling security, configuring a server separatation topology, and configuring advanced application server settings.

"Deploy and start a new application" on page 61
This topic describes how to deploy an application into your application server. It also provides information on uninstalling applications and configuring application settings.

Tune performance
This topic provides information about monitoring and tuning performance for your application server and applications.

"Backup and recovery considerations for WebSphere Application Server - Express" on page 73
This topic provides information on backing up and restoring data and configurations for your applications and your application server.

"Reference" on page 81
This topic provides reference information for the administrative tasks and tools.

Administration considerations for preinstalled systems

If WebSphere Application Server - Express is preinstalled on your iSeries server, an application server and an HTTP server are preconfigured for you. The application server is called ITD and includes the deployed IBM Telephone Directory V5.2 and IBM Welcome Page V1.1 enterprise Web applications provided exclusively for iSeries servers. The HTTP server is also called ITD and is configured for the ITD application server.

IBM Telephone Directory and IBM Welcome Page are enterprise Web applications provided by IBM Business Solutions. The applications are easy to understand, simple to use, provide services for any business, and show the use and integration of Web technologies on iSeries. See the following for more information:

- IBM Business Solutions

  Web site
- IBM Business Solutions documentation

Verify that the HTTP server and applcation server exist

To verify that the ITD HTTP server and the ITD application server exist, follow these steps:

1. Start the HTTP Server Administration interface.
2. In the **Server** list, look for these entries:
   - ITD - Apache
   - ITD - WAS - Express V5

   If these entries exist, the HTTP server and the application server are preconfigured.

Note: By default, the preconfigured ITD application server uses a block of ports beginning with port 2030 (ports 2030-2041). If another process on your iSeries server uses these ports already, you must change the ports for the ITD application server. For information on changing the ports assigned to an application server, see "Change application server ports with the chgwassvr script" on page 50 and "Change application server ports with the console and wsadmin" on page 52.

Install and run other applications on a preinstalled system

On a preinstalled system, you do not need to create a new application server before you deploy applications. You can deploy your applications in the ITD application server.

- If you want to deploy applications into the ITD application server, you can continue directly to "Configure the application server to run your application" on page 15.
- If you want to create a new application server for your applications, see "Create a new application server."

Create a new application server

An application server provides the runtime environment for your applications. An instance of WebSphere Application Server - Express consists of a single application server, which connects to an HTTP server instance to receive client requests. The application server performs administrative functions and provides services that your application uses to process client requests.

To create a new application server with HTTP Server Administration interface, follow these steps:

1. Start the HTTP Server Administration interface.
2. Click the **Setup** or **Manage** tab.

3. Expand **Tasks and Wizards**.
4. Click **Create New Express Server**.

   The Create New Express Server wizard creates a new application server to use Web applications with dynamic content. Before you continue, familiarize yourself with virtual hosts, applications, data sources, and JDBC providers. These basic features are briefly explained in the wizard and the help text. To learn more, see the WebSphere Application Server for iSeries Version 5.0 Information Center.
5. Click **Next**.
6. Specify a unique name for the new application server.
7. Click **Next**.
8. Select the HTTP server type for your new application server to use.
   - **Create a new HTTP Server (powered by Apache)**
     Select this option to create a new HTTP Server (powered by Apache) instance.
     a. Click **Next**.
     b. Complete the Create a new HTTP Server (powered by Apache) form.
     c. Click **Next**. Continue to step 9.
   - **Select an existing HTTP Server (powered by Apache)**
     Select this option to use an existing HTTP Server (powered by Apache) instance.
     a. Click **Next**.
     b. Select an existing HTTP Server (powered by Apache) from the list.
     c. Click **Next**. Continue to step 9.
   - **Select an existing Domino HTTP server**
     Select this option to use an existing Domino HTTP server.
     a. Click **Next**.
     b. Select an existing Domino HTTP server from the list.
     c. Click **Next**. Continue to step 9.
9. Specify the first port in a block of 12 unused ports on your system. The wizard assigns these ports to internal services of the application server. For example, if you specify 3001 as the first port, the wizard configures ports 3001 to 3012.
10. Click **Next**.
11. Install one or more of the example applications that are provided. Use the installed example applications to verify the application server is working correctly.
12. Click **Next**.
13. The summary page lists all of the choices you have made in the wizard. If any of the information displayed is incorrect, click **Back** until you reach the wizard form with the incorrect information and make your corrections. Click **Finish** to complete the wizard.

After you create your application server, continue to “Start and test your application server” on page 8.

For more information on working with application server instances, see these topics:

“Delete an application server instance” on page 4
This topic describes how to delete an application server instance with the HTTP Server Administration interface.

“Create an application server instance with the crtwasinst script” on page 4
This topic describes how to use the crtwasinst script in Qshell to create a new application server instance.

"Delete an application server instance with the dltwasinst script" on page 7
This topic describes how to use the dltwasinst script in Qshell to remove an application server instance.

"Display application server properties with the dspwasinst script" on page 7
This topic describes how to use the dspwasinst script in Qshell to view information about an application server instance.

Configure a new HTTP server instance
This topic describes how to configure an instance of IBM HTTP Server (powered by Apache) for your application server instance.

Configure Lotus Domino Web server
This topic describes how to configure an instance of Lotus Domino Web server for your application server instance.

Delete an application server instance

You may want to remove an application server instance if it is no longer needed. To remove an application server from the HTTP Server Administration interface, follow these steps:

1. Start the HTTP Server Administration interface.
2. From the **Server** list, select **All servers**.
3. Select the application server that you want to remove.
4. Click **Delete**.
5. Click **OK**.

Create an application server instance with the crtwasinst script

You can use the crtwasinst script from Qshell to create an application server instance. You can create multiple WebSphere Application Server - Express instances that are completely isolated from one another. For example, you can create separate instances for application development and application testing, or you can create one instance with security enabled and one with security disabled.

The crtwasinst script creates a new instance that contains one application server. It also creates the required directories and sets up the correct authorities. For information on running Qshell scripts, see "Configure Qshell to run WebSphere Application Server - Express scripts" on page 92.

Authority

To run this script, your iSeries user profile must have *ALLOBJ authority.

Usage

To create a new instance with the crtwasinst script, follow these steps:

1. On the CL command line, enter the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the crtwasinst script:
   ```
   crtwasinst -instance instance -portblock port
   ```
   where *instance* is the name of the instance that is created and *port* is the first of a block of ports.

Syntax

The syntax of the script is shown below.

```
crtwasinst -instance instance [ -portblock portblock ]
 [ -server servername ] [ -exthttp exthttpport ]
 [ -inthttp inthttpport ] [ -admin adminport ] [ -adminssl adminsslport ]
 [ -soap soapport ] [ -nameservice nameserviceport ] [ -sas sasserverport ]
 [ -csiv2server csiv2serverauthport ] [ -csiv2client csiv2clientauthport ]
 [ -verbose ] [ -help ]
```

Parameters

The parameters of the script are:

- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance. The script creates the new instance in the /QIBM/UserData/WebASE/ASE5/*instance* directory.
- **-portblock**
  This is an optional parameter. The value *portblock* specifies the first number of a block of port numbers that your instance uses. Specify the first port in a group of unused ports on your iSeries server. You can use the Work with TCP/IP Network Status (NETSTAT *CNN) command to display a list of port numbers that are currently being used.

  Note:

  1. A WebSphere Application Server - Express instance uses several ports for various functions. When you create a new instance, ports are assigned based on the following ordered conditions:
     a. **Specific port parameters**
        If you specify values for specific port parameters, the script uses those values. Specific port parameters are -inthttp, -admin, -soap, -nameservice, and -drsclient.
     b. **The -portblock parameter**
        Services for which you have not specified a port number are assigned ports sequentually starting with the value of the -portblock parameter. If a script encounters a port that is assigned to a different instance or a port specified by another parameter in the script, it skips that port number and continues with the next unused port.
     c. **Default values**
        If -portblock is not specified, any services for which you have not specified a port parameter are assigned the default ports. See the specific port parameters for the default port numbers.
- **-server**
  This is an optional parameter. The value *servername* specifies the name of the application server that runs in your instance. If this value is not specified, the application server name is the same as the instance name.
- **-exthttp**
  This is an optional parameter. The value *exthttpport* specifies the number of the TCP/IP port where the external HTTP server listens. The default value is 80. You must configure the external HTTP server through its administrative interface so that WebSphere Application Server - Express also listens on this port.

  Note: The -portblock parameter does not affect the external HTTP port. If you specify the -portblock parameter, but not the -exthttp parameter, your instance uses the default value for the external HTTP port.
- **-inthttp**
  This is an optional parameter. The value *inthttpport* specifies the port number on which the Web container listens for requests from the Web server. If neither the -portblock parameter nor the -inthttp parameter is specified, the default value is 9080. See the Note (page 5) on the -portblock parameter for more information.
- **-admin**
  This is an optional parameter. The value *adminport* specifies the port number to use for the WebSphere administrative console. If neither the -portblock parameter nor the -admin parameter is specified, the default value is 9090. See the Note (page 5) on the -portblock parameter for more information.

- **-adminssl**
  This is an optional parameter. The value *adminportssl* specifies the port number to use for the secure communications with WebSphere administrative console. If neither the -portblock parameter nor the -adminssl parameter is specified, the default value is 9043. See the Note (page 5) on the -portblock parameter for more information.
- **-soap**
  This is an optional parameter. The value *soapport* specifies the port number to use for Simple Object Access Protocol (SOAP). If neither the -portblock parameter nor the -soap parameter is specified, the script assigns the default value. The default value is 8880. See the Note (page 5) on the -portblock parameter for more information.
- **-nameservice**
  This is an optional parameter. The value *nameserviceport* specifies the port number to use for name service (or RMI connector) port. If neither the -portblock parameter nor the -nameservice parameter is specified, the script assigns the default value. The default value is 2809. See the Note (page 5) on the -portblock parameter for more information.
- **-sas**
  This is an optional parameter. The value *sasserverport* specifies the port on which the Secure Association Services (SAS) listen for inbound authentication requests. The default value is 9401. This port is specified by the SAS_SSL_SERVERAUTH_LISTENER_ADDRESS property in serverindex.xml.

  Note: It is recommended that you specify this parameter. If you do not specify this parameter, the application server selects a port at runtime. However, if a client application is connected to the application server and the application server restarts, the server could select a different port number, and the client application would be unable to connect to the server.
- **-csiv2server**
  This is an optional parameter. The value *csiv2serverauthport* specifies the port on which the Common Secure Interoperability Version 2 (CSIV2) Service listens for inbound server authentication requests. The default value is 9403. This port is specified by the CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS property in serverindex.xml.

  Note: It is recommended that you specify this parameter. If you do not specify this parameter, the application server selects a port at runtime. However, if a client application is connected to the application server and the application server restarts, the server could select a different port number, and the client application would be unable to connect to the server.
- **-csiv2client**
  This is an optional parameter. The value *csiv2clientauthport* specifies the port on which the Common Secure Interoperability Versson 2 (CSIV2) Service listens for inbound client authentication requests. The default value is 9402. This port is specified by the CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS property in serverindex.xml.

  Note: It is recommended that you specify this parameter. If you do not specify this parameter, the application server selects a port at runtime. However, if a client application is connected to the application server and the application server restarts, the server could select a different port number, and the client application would be unable to connect to the server.
- **-verbose**
  This optional parameter turns on verbose messages, which can be helpful if you need to debug the script.
- **-help**
  This optional parameter displays the help message. If you specify this parameter, the script ignores all other parameters.

Example

In this example, the script creates an instance named devinst.

```
crtwasinst -instance devinst -portblock 10320
```

Delete an application server instance with the dltwasinst script

The dltwasinst script removes an instance and the files associated with it. For information on running Qshell scripts, see "Configure Qshell to run WebSphere Application Server - Express scripts" on page 92.

Authority

To run this script, your iSeries user profile must have *ALLOBJ authority.

Usage

To run the dltwasinst script, follow these steps:
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the dltwasinst script:
   ```
   dltwasinst -instance instance
   ```
   where *instance* is the name of the instance that you want to delete.

Syntax

The syntax of the script is:
```
dltwasinst -instance instance [ -verbose ] [ -help ]
```

Parameters

The parameters of the script are:
- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance that you want to delete.
- **-verbose**
  This optional parameter turns on verbose messages, which can be helpful if you need to debug the script.
- **-help**
  This optional parameter displays the help message. If you specify this parameter, the script ignores all other parameters.

Example

In this example, the instance testinst is deleted.
```
dltwasinst -instance testinst
```

Display application server properties with the dspwasinst script

The dspwasinst script displays information about an application server instance. The script displays this information:
- Application servers. For each application server, the following information is displayed:
  – Port numbers
  – Installed applications
  – Status (running or stopped)
  – Job ID (if the application server is running)
- Cell name
- Node name

For information on running Qshell scripts, see “Configure Qshell to run WebSphere Application Server - Express scripts” on page 92.

Authority

To run this script, your iSeries user profile must have *ALLOBJ authority.

Usage

To display information for an instance and for the application servers in that instance, run the dspwasinst script from the Qshell command line. To run the dspwasinst script, follow these steps:
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the dspwasinst script:
   ```
   dspwasinst -instance instance
   ```
   where *instance* is the name of the instance that you want to display.

Syntax

The syntax of the script is:
```
dspwasinst -instance instance [ -server servername ] [ -help ]
```

Parameters

The parameters of the script are:
- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance that you want to display the properties for.
- **-server**
  This is an optional parameter. The value *servername* specifies the name of the application server to display. If this value is not specified, all of the application servers in the instance are displayed. To display properties for multiple servers, specify multiple -server parameters.
- **-help**
  This optional parameter displays the help message. If you specify this parameter, the script ignores all other parameters.

Example

In this example, the script displays information about the instance devinst.
```
dspwasinst -instance devinst
```

Start and test your application server

After you complete the Create New Express Server wizard, run the ExpressSamples application to verify that the application server is running correctly.

Note: Before you test your application server, ensure the HTTP Server (powered by Apache) or the Domino HTTP server you selected the application server to use is *Running*. See Manage HTTP Servers for more information on HTTP Server (powered by Apache). See the Lotus(R) Domino(TM) Web site

for more information on Domino HTTP servers.

To start and test a new application server or to test a new application, follow these steps:

1. Start the HTTP Server Administration interface.
2. Click the **Manage** tab.
3. Select an application server from the **Server** list.
4. Click the **Start** button (

). The start button is located directly below the **Server** list and to the right of the application server status message. The status message changes from *Stopped* to *Starting*. When the server is ready, the status message changes to *Running*. If the status message changes back to *Stopped*, an error has occurred.

   Note: If the status message for the application server does not change over a short period of time, click the **Refresh** button (

). To stop your server, click the **Stop** button (

).
5. Expand **Applications** after your application is running.
6. Click **Manage Installed Applications**.
7. Ensure the application you want to test is running. If it is not, select the application from the installed applications list and click **Start**. See "Manage installed applications for your application server" on page 62 for more information.
8. Open a new Web browser window.
9. Specify one of the following URLs to test your application server, where *your.server.name* is the name of your iSeries server and *port* is the port number of your HTTP Server instance. Use the example application you installed.
   - **ExpressSamples**
     http://*your.server.name*:*port*/Snoop
   - **DB2 Web Services**
     http://*your.server.name*:*port*/services

Note: You can also use these steps to test new applications. After you "Deploy and start a new application" on page 61, specify the URL in a Web browser. For example, if you install the application *myApp* on the server *my.iSeries.server*, and your HTTP Server instance uses port *10080*, specify this URL:

```
http://my.iSeries.server:10080/myApp
```

After you verify that the application server is working correctly, "Configure the application server to run your application" on page 15.

For additional information on starting and stopping application servers, see these topics:

"Start an application server with the startServer script" on page 10
This topic describes how to use the startServer script in Qshell to start an application server.

"Use the CL command to start an application server" on page 11
This topic describes how to use a CL command on an CL command line to start an application server.

"Verify that the application server has started" on page 12
Before you can start your applications, the application server must be running. This topic describes how you can verify that the application server is running.

"Stop an application server with the stopServer script" on page 13
This topic describes how to use the stopServer script in Qshell to stop an application server.

"Display the status of your application server with the serverStatus script" on page 14
This topic describes how to use the serverStatus script in Qshell to determine the status of an application server.

Start an application server with the startServer script

The startServer script reads the configuration file for the specified server process and starts the server.

For information on running Qshell scripts, see "Configure Qshell to run WebSphere Application Server - Express scripts" on page 92.

Authority

To run this script, your user profile must have *ALLOBJ authority. For information on setting explicit authorities for a user profile that does not have *ALLOBJ authority, see "Set explicit authorities for the startServer and stopServer scripts" on page 94.

Usage

To start an application server with the startServer script, follow these steps:
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the startServer script:
   ```
   startServer -instance instance server
   ```
   where *instance* is the name of the instance that you want to start and *server* is the name of the server that you want to start.

When the application server is running, the script displays this message:
```
ASE6123: Application server started.
 Cause . . . . . :   Application server instance in ASE5 instance instance has
   started and is ready to accept connections on admin port admin_port
```

where *instance* is the name of your instance and *admin_port* is the port number for the WebSphere administrative console.

Note: If you specify the -nowait parameter, the script does not display this message. You must "Verify that the application server has started" on page 12 before you continue.

Syntax

The syntax of the startServer script is:
```
startServer server -instance instance
 [ -nowait ] [ -timeout seconds ] [ -trace ] [ -help | -? ]
```

where *server* is the name of the configuration directory of the server you want to start.

Parameters

The parameters of the startServer script are:

- ***server***
  This required parameter specifies the name of the server that you want to start. If *server* is not specified, the server name defaults to the value specified for the -instance parameter. This value is case sensitive.
- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance that contains the server that you want to start.
- **-nowait**
  This is an optional parameter. If you specify this parameter, the script returns control to the user without waiting for successful initialization of the server. The default is to wait for successful initialization.
- **-timeout**
  This is an optional parameter. The value *seconds* specifies the amount of time in seconds to wait for successful initialization of the server. The script returns control to the user at the end of the timeout value. The default is to wait until the server initialization is complete.
- **-trace**
  This is an optional parameter. If you specify this parameter, the trace function is turned on. The trace output is written to the JOBNAME.JOBUSER.JOBNUMBER file in the /QIBM/UserData/WebASE/ASE5/service/trace directory, where JOBNAME, JOBUSER, and JOBNUMBER are taken from the application server job. The job name is output when the job is submitted by the script.
- **-help** or **-?**
  This optional paramter prints the usage statement for the script.

Examples

```
startServer -instance myinst
```

This example starts the myinst application server in the myinst instance.

```
startServer devinst -instance devinst -nowait
```

This example starts the devinst application server in the devinst instance and returns control immediately to the user.

Use the CL command to start an application server

You can use a CL command to start your application server instance. To start the application server from the CL command line, follow these steps:

1. Start Transmission Control Protocol/Internet Protocol (TCP/IP). On the CL command line, enter this command:
   ```
   STRTCP
   ```
2. To start the QASE5 subsystem, run the following command on the CL command line:
   ```
   STRSBS QASE5/QASE5
   ```
   Note: Your user profile must have *JOBCTL authority to start the WebSphere Application Server subsystem.
3. After the subsystem starts, run this command from the CL command line:
   ```
   SBMJOB CMD(CALL PGM(QASE5/QASESTRSVR) PARM('-instance'
     '/QIBM/UserData/WEBASE/ASE5/instance' '-server' 'server'))
     JOB(server) JOBD(QASE5/QASE5) JOBQ(QASE5/QASE5) USER(QEJBSVR)
     LANGID(*USRPRF) CNTRYID(*USRPRF) CCSID(*USRPRF)
   ```
   where *instance* is the name of the application server instance that you want to start and *server* is the name of the server in that instance. For most WebSphere Application Server - Express instances, the

instance name and server name are the same. You can only start one application server when you run this command. To start additional application servers, you must run the command separately for each application server that you want to start.

Note: To run this command, your user profile must have *USE authority to the QEJBSVR user profile. Use the Edit Object Authority (EDTOBJAUT) command to add or verify that your user profile has this authority.

If your profile has authority to the QASE5/QASE5 job description and QASE5/QASE5 subsystem description, you can configure WebSphere Application Server - Express to automatically start when the you start the WebSphere Application Server - Express subsystem.

1. Create a duplicate of the job description used by WebSphere Application Server instances. For example, on the CL command line run this command:
   ```
   CRTDUPOBJ OBJ(QASE5) FROMLIB(QASE5) OBJTYPE(*JOBD) TOLIB(mywasjobd) NEWOBJ(myserv)
   ```
2. Use the CHGJOBD command to change the newly created job description so that the **Request data or command** (RQSDTA) field starts the server. In the following example, the instance is myinst and the server is myserv:
   ```
   'QSYS/CALL PGM(QASE5/QASESTRSVR) PARM(''-instance''
     ''/QIBM/UserData/WebASE/ASE5/myinst'' ''-server'' ''myserv'')'
   ```
3. Add an autostart job entry to the QASE5/QASE5 subsystem. Enter this command from the CL command line:
   ```
   ADDAJE SBSD(QASE5/QASE5) JOB(myserv) JOBD(mywasjobd/myserv)
   ```

Verify that the application server has started

When the WebSphere Application Server - Express environment is ready for use, a message is written to the job log of the application server job indicating that the WebSphere Application Server environment is ready.

To verify that your application server instance has started successfully, follow these steps:

1. On an CL command line, run the Work with Active Jobs (WRKACTJOB) command.
2. Find your application server job in the QASE5 subsystem.
3. Specify option 5 (Work with Job) on the option line next to the job, and press **Enter**.
4. On the command line of the Work with Job display, specify option 10 (Display job log, if active), and press **Enter**.
5. Press **F10** to display all messages.
6. Look for this message:
   ```
   WebSphere application server application_server ready.
   ```
   where *application_server* is the name of your application server.

   If the message is not displayed, press **F5** to refresh the job log messages until the message is displayed. When the message is displayed, the WebSphere Application Server environment has successfully started. It may take up to 20 minutes for the message to be displayed, depending on your iSeries server. If the message is not displayed, see WebSphere Application Server startup in the Troubleshooting section.
7. To display the port number on which the application server is listening for the administrative console, position the cursor on the last line of the message and press **F1**. This message is displayed:
   ```
   WebSphere application server application_server
    in job app_server_job is ready to handle administrative requests
    on port port_number.
   ```
   where *application_server* is the name of your application server, *app_server_job* is the i5/OS job name for your application server, and *port_number* is the number of the port used by the administrative console.
8. Press **F3** twice to exit.

Stop an application server with the stopServer script

The stopServer script stops the specified server. For information on running Qshell scripts, see "Configure Qshell to run WebSphere Application Server - Express scripts" on page 92.

Authority

To run this script, your user profile must have *ALLOBJ authority. For information on setting explicit authorities for a user profile that does not have *ALLOBJ authority, see "Set explicit authorities for the startServer and stopServer scripts" on page 94.

Usage

To stop an instance with the stopServer script, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the stopServer script:
   ```
   stopServer -instance instance server
   ```
   where *instance* is the name of the instance you want to stop and *server* is the name of the application server that you want to stop.

Syntax

The syntax of the stopServer script is:

```
stopServer server -instance instance [ -nowait ] [ -quiet ]
 [ -logfile filename ] [ -replacelog ] [ -trace ] [ -timeout seconds ]
 [ -statusport statusportnumber ] [ -port portnumber ]
 [ -username username ] [ -password password ] [ -conntype SOAP | RMI ]
 [ -help | -? ]
```

Parameters

The parameters of the stopServer script are:

- ***server***
  This is a required parameter. The value *server* specifies the name of the server that you want to stop. This value is case sensitive.
- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance that contains the server that you want to stop.
- **-nowait**
  This is an optional parameter. If you specify this parameter, the script returns control to the user without waiting for the server to stop successfully. The default is to wait for the server to stop successfully.
- **-quiet**
  This is an optional parameter. If you specify this parameter, the script does not display informational messages. The default is to display informational messages while the script runs.
- **-logfile**
  This is an optional parameter. The value *filename* specifies the location and name of the log file for the script. The default value is /QIBM/UserData/WebASE/ASE5/*instance*/logs/*server*/stopServer.log where *instance* is the name of the instance that contains the server that you want to stop, and *server* is the name of the server that you want to stop.

- **-replacelog**
  This is an optional parameter. If you specify this parameter, the script replaces the log file if it exists. By default the script appends to the log file if it exists.
- **-trace**
  This is an optional parameter. If you specify this parameter, the script outputs additional trace information to the log file for the script. You should only specify this parameter if errors occur when you try to stop a server. The default is to not log additional trace information.
- **-timeout**
  This is an optional parameter. The value *seconds* specifies the amount of time in seconds to wait for the server to stop before returning control to the caller. The default is to wait until the server has stopped successfully.
- **-statusport**
  This is an optional parameter. The value *statusportnumber* specifies the port on which to listen for the status of the server while it is stopping. The default is to use the next available port.
- **-port**
  This is an optional parameter. The value *portnumber* specifies the SOAP or RMI port for the server. If you specify this parameter, the stopServer script sends the stop command directly to server. If you specify the RMI port value for this parameter, you must also specify the -conntype parameter. By default, the script reads the configuration files to obtain the information that is necessary to stop the server.
- **-conntype**
  This is an optional parameter. If you specify the -port parameter, the -conntype parameter specifies the connector type to use. Valid values are SOAP or RMI. The default value is SOAP.
- **-username**
  This parameter is required if security is enabled for the server. The value *username* specifies the user name for authentication.
- **-password**
  This parameter is required if security is enabled for the server. The value *password* specifies the password for authentication.
- **-help** or **-?**
  This optional paramter prints the usage statement for the script.

Example

```
stopServer myserver -instance myserver -port 10380
```

This example uses port 10380 to stop the myserver application server in the myserver instance.

Display the status of your application server with the serverStatus script

Use the serverStatus script in Qshell to obtain the status of an application server instance. For information on running Qshell scripts, see “Configure Qshell to run WebSphere Application Server - Express scripts” on page 92.

Authority

No special authority is required to run this script.

Usage

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the serverStatus script:

```
serverStatus server
```

where *server* is the name of your application server.

Syntax

The syntax of the serverStatus script is:

```
serverStatus server | -all -instance instance
 [ -logfile filename ] [ -replacelog ] [ -trace ] [ -username username ]
 [ -password password ] [ -help | -? ]
```

Parameters

The parameters of the serverStatus script are:

- ***server***
  The value *server* is name of the server for which you want to display status. You must specify either a server name or the -all parameter. This value is case sensitive.
- **-all**
  If you specify this parameter, the script displays the status of all of the servers in the instance. You must specify either the -all parameter or a server name.
- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance that contains the server or servers for which you want to display status.
- **-logfile**
  This is an optional parameter. The value *filename* specifies the location and name of the log file for the script. The default value is /QIBM/UserData/WebASE/ASE5/*instance*/logs/serverStatus.log where *instance* is the name of the instance that contains to which the server for which you want to display status.
- **-replacelog**
  This is an optional parameter. If you specify this parameter, the script replaces the log file if it exists. By default the script appends to the log file if it exists.
- **-trace**
  This is an optional parameter. If you specify this parameter, the script outputs additional trace information to the log file for the script. You should only specify this parameter if errors occur when you try to display server status. The default is to not log additional trace information.
- **-username**
  This parameter is required if security is enabled for the server. The value *username* specifies the user name for authentication.
- **-password**
  This parameter is required if security is enabled for the server. The value *password* specifies the password for authentication.
- **-help** or **-?**
  This optional paramter prints the usage statement for the script.

Example

```
serverStatus myAppSvr
```

This example displays the status for the myAppSvr instance.

Configure the application server to run your application

After you create an application server and test it, you must configure it for your application. These topics describe how to configure application server settings and resources for your application.

"Configure classloaders for your application server"
Classloaders determine how your application server loads Java class files. This topic describes how to use the administrative console to configure classloaders for your application server.

"Administer HTTP transports for the Web container with the administrative console" on page 17
Components of your application server use HTTP to communicate with each other. This topic describes how use the administrative console and wsadmint to confiugre HTTP transports for your application server.

"Administer session tracking for your application server" on page 23
Applications use sessions to keep track of the client requests from a single Web browser session. This topic describes how to use the administrative console and wsadmin to configure session tracking for your application server.

"Manage substitution variables with the administrative console" on page 24
Substitution variables can be useful in writing scripts and configuring classloaders. This topic describes how to use the administrative console and wsadmin to set up variables for your application server.

"Manage virtual hosts for your application server" on page 26
Virtual hosts allow you to configure multiple multiple host names and port numbers as a single logical host. This topic describes how to configure virtual hosts for your application server instance.

"Configure database access for your application" on page 31
Applications use JDBC drivers to connect to databases. JDBC drivers are represented by JDBC providers, and databases are represented by data sources. This topic describes how to configure JDBC providers and data sources for your application server.

"Administer mail resources" on page 39
Your application server includes a built-in JavaMail$^{(TM)}$ provider that supports Java-based e-mail client applications. This topic describes how to use the administrative console and wsadmin to configure mail resources for your application server.

"Regenerate the Web server plugin configuration" on page 41
Application servers use a Web server plugin to communicate with your HTTP server instance. Some changes to your application server require you to update the plugin configuration file. This topic describes how to regenerate the plugin configuration, and the situations in which you need to do so.

"Configure a remote HTTP topology" on page 43
This topic describes how to configure a remote HTTP topology, in which your application server and your HTTP server are hosted on separate machines or logical partitions.

"Configure security settings for your application server" on page 44
This topic provides some information on configuring security for your application server instance. For more detailed information on configuring security for WebSphere Application Server - Express, see Security.

"Advanced application server settings" on page 49
This topic describes advanced settings for your application server. Under most circumstances, you do not need to change these advanced settings.

Configure classloaders for your application server

Classloaders determine how your application server loads Java class files. For more information, see the Classloaders topic in *Application Development*.

1. "Start the WebSphere administrative console" on page 83.

2. Expand **Servers** and click **Application Servers**.
3. Click the name of your application server.
4. On the application server's page, click **Classloader**.
5. Add, modify, or remove a classloader.
   - To add a new classloader for the application server, follow these steps:
     a. Click **New**.
     b. Select a classloader mode.
     c. Click **Apply** or **OK**.
   - To modify a classloader, follow these steps:
     a. Click the classloader that you want to modify.
     b. Make your changes.
     c. Click **Apply** or **OK**.
   - To remove a classloader, follow these steps:
     a. Select the check box for the classloader that you want to remove.
     b. Click **Delete**.
     c. Click **Apply** or **OK**.
6. "Save the application server configuration" on page 84.

Administer HTTP transports for the Web container with the administrative console

Components of WebSphere Application Server - Express uses HTTP transports to communicate with each other. Your application also uses HTTP transports to receive and reply to client requests.

Before you decide which type of transport to use, consider the following limitations of the internal HTTP transports:

- Internal transports do not support Simple Network Management Protocol (SNMP).
- Existing Common Gateway Interface (CGI) support must be implemented as servlets. That is, you must convert any existing CGI programs to servlets.
- Internal transports do not support session affinity for workload balancing.
- Because static content can not be served directly by an external HTTP server, overall application server performance can be adversely affected.
- Web servers generally are more susceptible to security exposures than application servers are. If a production environment includes an external Web server and an application server running on two separate machines, many of the Web server's security exposures can be limited to the machine where the Web server runs. However, if you use the internal HTTP transport, the internal HTTP transport always runs on the same machine as the application server. As a result, a security exposure in the Web server can expose both the Web server's and application server's environments.

You can use the administrative console and wsadmin to configure HTTP transports for your application server.

- Use the administrative console to configure HTTP transports (page 17)
- Use wsadmin to configure HTTP transports (page 18)

Use the administrative console to configure HTTP transports

1. "Start the WebSphere administrative console" on page 83.
2. Expand **Servers** and click **Application Servers**.
3. Click the name of your application server.
4. On the application server's page, click **Web Container**.

5. On the Web Container page, click **HTTP transports**.
6. Add, modify, or remove an HTTP transport.
   - To add a new HTTP transport, follow these steps:
     a. Click **New**.
     b. Configure the HTTP transport settings.
     c. Click **Apply** or **OK**.
   - To modify an HTTP transport, follow these steps:
     a. Click the host name for the transport that you want to modify.
     b. Make your changes.
     c. Click **Apply** or **OK**.
   - To remove an HTTP transport, follow these steps:
     a. Select the transport that you wanto to remove.
     b. Click **Delete**.
7. "Save the application server configuration" on page 84.

Use wsadmin to configure HTTP transports

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the application server and assign it to the server variable:
   ```
   set server [$AdminConfig getid /Cell:myCell/Node:myNode/Server:myAppSvr/]
   ```
   where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, and *myAppSvr* is the name of your application server.
5. Run this command to identify the Web container for the application server and assign it to the wc variable:
   ```
   set wc [$AdminConfig list WebContainer $server]
   ```
6. Run these commands to list all of the transports that belong to the Web container and assign it to the transports variable:
   ```
   set transportsAttr [$AdminConfig showAttribute $wc transports]
   set transports [lindex $transportsAttr 0]
   ```
   These commands return the transport objects from the transports attribute in a list format.
7. Run this command to identify the transport to be modified and assign it to the transport variable:
   ```
   set transport [lindex $transports 0]
   ```
8. Run this command to modify the address attribute to change the host and port:
   ```
   $AdminConfig modify $transport {{address {{host {myHost}} {port 9081}}}}
   ```
9. Run this command to save your changes:
   ```
   $AdminConfig save
   ```

Set custom properties for an HTTP transport

Several HTTP transport properties are not shown in the administrative console settings page for an HTTP transport. To specify values for these custom properties for a specific transport on the HTTP transport **Custom Properties** page, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Servers** and click **Application Servers**
3. Click the name of your application server.
4. On the application server page, click **Web Container**.

5. On the **Web Container** page, click **HTTP Transports**.
6. Click the host whose properties you want to set.
7. Under **Additional Properties**, click **Custom Properties**.

   Note: You can also set these properties on the Web Container Custom Properties page.
8. On the **Custom Properties** page, click **New**.
9. On the settings page for a new property, type the name of the transport property and the value that you want to set for that property. For example, if you want the transport to wait a maximum of 60 seconds when trying to read or write data during a request, type `ConnectionIOTimeout` for the name and `60` for the value.
10. After you specify each property, click **OK**.
11. "Save the application server configuration" on page 84.
12. Restart the server.
13. "Regenerate the Web server plugin configuration" on page 41.

You can add any of these custom properties to manage HTTP transports:

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

  This value determines how long the application server waits while receiving two subsequent data packets for the same HTTP request. For example, with the default ConnectionIOTimeout setting of five seconds, if an HTTP client sends two data packets spaced six seconds apart, the process times out, and the server throws a java.io.InterruptedIOException error. The server terminates the HTTP request, and the HTTP client must resubmit the request. The default value is 5 seconds.

  Note: When a client attempts to send a large amount of request data, such as a file upload, there is a greater possibility of an InterruptedIOException at the application server. To avoid this problem, you might want to increase the ConnectionIOTimeout value for the Web container.
- **ConnectionKeepAliveTimeout**
  Specifies the maximum number of seconds to wait for the next request on a keep-alive connection. The default value is 5 seconds.
- **ConnectionResponseTimeout**
  Specifies the maximum number of seconds to wait when trying to read or write data during a response. The default value is 300.
- **KeepAliveEnabled**
  Specifies whether to keep connections alive or not. The default value is `true`.

  You can set these properties on either the Web Container or HTTP Transport Custom Properties pages. When set on the Web container Custom Properties page, all transports inherit the properties. Setting the same properties on a transport overrides like settings defined for a Web container.
- **MaxKeepAliveConnections**
  Specifies the maximum number of concurrent keep-alive (persistent) connections across all HTTP transports. To make a particular transport close connections after a request, you can set MaxKeepAliveConnections to `0` (zero) or you can set KeepAliveEnabled to `false` on that transport.

  The Web server plug-in keeps connections open to the application server as long as it can. However, if the value of this property is too small, performance is negatively impacted because the plug-in has to open a new connection for each request instead of sending multiple requests through one connection. The application server might not accept a new connection under a heavy load if there are too many sockets in TIME_WAIT state. If all client requests are going through the Web server plug-in and there are many TIME_WAIT state sockets for port 9080, the application server is closing connections prematurely, which decreases performance. The application server closes the connection from the plug-in, or from any client, for any of the following reasons:
  - The client request was an HTTP 1.0 request when the Web server plug-in always sends HTTP 1.1 requests.

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

  The default value is 90% of the maximum number of threads in the Web container thread pool. This prevents all of the threads from being held by keep alive connections so that there are threads available to handle new incoming connect requests.
- **MaxKeepAliveRequests**
  Specifies the maximum number of requests which can be processed on a single keep alive connection. This parameter can help prevent denial-of-service attacks when a client tries to hold on to a keep-alive connection. The Web server plug-in keeps connections open to the application server as long as it can, providing optimum performance. The default value is 100.
- **MutualAuthCBindCheck**
  Specifies whether or not a client certificate should be resolved to a SAF principal. The default value is `false`. If you set this property to `true`, all SSL connections from a client must have a client certificate, and the user ID associated with the client certificate must have RACF CONTROL authority for CB.BIND.*servername*. If the client request does not meet these conditions, the connection is closed. To grant the certificate's user ID RACF CONTROL authority, run this command:
  ```
  PERMIT CB.BIND.servername CLASS(CBIND) ID(userID) ACCESS(CONTROL)
  ```
  where *servername* is the name of your application server and *userID* is the user ID that is associated with the client certificate.
- **protocol_http_large_data_inbound_buffer**
  Specifies the length, in bytes, of a serially reusable inbound buffer. The transport uses this buffer for HTTP requests that are larger than 10MB. The default value is 0. A value of 0 specifies that no buffer is needed, and the transport rejects client requests that are larger than 10MB.
- **TrustedProxy**
  Specifies whether or not the transport trusts Private Headers from a WebSphere Application Server - Express plug-in for a Web server.
- **AccessLogDisable** and **AccessLog**
  These properties specify access logging settings. For more information, see "Configure access logging for internal Web server HTTP transport" on page 22.
- **ErrorLogDisable**, **ErrorLog**, and **LogLevel**
  These properties specify error logging settings. For more information, see "Configure logging for internal Web server HTTP transport."

Configure logging for internal Web server HTTP transport

An independent logging mechanism is available to monitor the HTTP transport. To configure logging for internal HTTP transports, add these custom properties to one of the HTTP transports in your Web container:

Property name	Valid Values	Description	Default	Scope
ErrorLogDisable	true/false	Enable or disable logging	true (logging disabled)	Virtual/Global
ErrorLog	*filename*	The location of the log file	logs/*app_server*/http.log[1]	

Property name	Valid Values	Description	Default	Scope
LogLevel	• debug [2] • info • warn • error • crit	The level of messages to log	warn	Virtual/Global

[1] Relative paths are qualified from the root directory of your instance. For example, for the default application server in the default instance of WebSphere Application Server - Express, the default location for the log file is /QIBM/UserData/WebAS5/Base/default/logs/server1/http.log.

[2] Each level also logs messages for the levels below it. For example, the warn level logs warn, error, and crit messages. The debug level logs all messages.

Enable HTTP transport logging

To enable HTTP transport logging, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Servers** and click **Application Servers**.
3. Click the name of the application server for which you want to disable HTTP transport logging.
4. On the application server page, click **Web Container** under **Additional Properties**.
5. On the **Web Container** page, click **HTTP Transports**.
6. On the **HTTP Transports** page, select one of the HTTP transports.
7. On the transport page, click **Custom Properties**.
8. To enable error logging, add the ErrorLogDisable custom property to one of the HTTP transports in your Web container and set the value to **false**:
   a. Click **New**.
   b. For the property name, specify `ErrorLogDisable`.
   c. For the value, specify `false`.
   d. Click **OK**.
9. (Optional) Specify the ErrorLog custom property:
   a. Click **New**.
   b. For the property name, specify `ErrorLog`.
   c. For the value, specify a file name.
   d. Click **OK**.
10. (Optional) Specify the LogLevel custom property:
    a. Click **New**.
    b. For the property name, specify `LogLevel`.
    c. For the value, specify the level of logging you want to record.
    d. Click **OK**.
11. "Save the application server configuration" on page 84.
12. Restart the application server.

Disable HTTP transport logging

To disable HTTP transport logging, follow these steps:

1. "Start the WebSphere administrative console" on page 83.

2. In the topology tree, expand **Servers** and click **Application Servers**.
3. Click the name of the application server for which you want to disable HTTP transport logging.
4. On the application server page, click **Web Container** under **Additional Properties**.
5. On the **Web Container** page, click **HTTP Transports**.
6. On the **HTTP Transports** page, click the transport that includes the ErrorLogDisable property.
7. On the transport page, click **Custom Properties**.
8. Click the ErrorLogDisable custom property.
9. On the property page, specify a valu of `true` for the ErrorLogDisable property.
10. "Save the application server configuration" on page 84.
11. Restart the application server.

Configure access logging for internal Web server HTTP transport

To log information about client HTTP requests, the internal Web server HTTP transport supports access logging. To configure access logging, add these custom properties to one of the HTTP transports in your Web container:

Property name	**Valid Values**	**Description**	**Default**	**Scope**
AccessLogDisable	true/false	Enable or disable access logging	true (logging disabled)	Virtual/Global
AccessLog	*filename*	The location of the log file	logs/*app_server*/http_access.log[1]	

[1] Relative paths are qualified from the root directory of your instance. For example, for the default application server in the default instance of WebSphere Application Server - Express, the default location for the log file is /QIBM/UserData/WebAS5/Base/default/logs/server1/http_access.log.

Access log entries should have this format:

```
hostname user_agent [local_time -status_code]
  thread_id http_request status_code bytecount
```

where:

- *hostname* is the name or IP address of the client
- *user_agent* is the type of application (for example, Internet Explorer or Netscape) that the client uses to connect to your application server
- *local_time* is the time that the server receives the request
- *status_code* is the status of the incoming request
- *thread_id* is the ID of the thread that processes the request
- *http_request* is the URL that the client requests
- *bytecount* is the number of bytes contained in the body of the request

Enable access logging

To enable access logging, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Servers** and click **Application Servers**.
3. Click the name of the application server for which you want to disable HTTP transport logging.
4. On the application server page, click **Web Container** under **Additional Properties**.
5. On the **Web Container** page, click **HTTP Transports**.
6. On the **HTTP Transports** page, select one of the HTTP transports.

7. On the transport page, click **Custom Properties**.
8. To enable error logging, add the AccessLogDisable custom property to one of the HTTP transports in your Web container and set the value to **false**:
   a. Click **New**.
   b. For the property name, specify `AccessLogDisable`.
   c. For the value, specify `false`.
   d. Click **OK**.
9. (Optional) Specify the AccessLog custom property:
   a. Click **New**.
   b. For the property name, specify `AccessLog`.
   c. For the value, specify a file name.
   d. Click **OK**.
10. "Save the application server configuration" on page 84.
11. Restart the application server.

Disable access logging

To disable access logging, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Servers** and click **Application Servers**.
3. Click the name of the application server for which you want to disable HTTP transport logging.
4. On the application server page, click **Web Container** under **Additional Properties**.
5. On the **Web Container** page, click **HTTP Transports**.
6. On the **HTTP Transports** page, click the transport that includes the AccessLogDisable property.
7. On the transport page, click **Custom Properties**.
8. Click the ErrorLogDisable custom property.
9. On the property page, specify a valu of `true` for the AccessLogDisable property.
10. "Save the application server configuration" on page 84.
11. Restart the application server.

Administer session tracking for your application server

You can use the administrative console and wsadmin to configure session tracking for your application server. In simple terms, an HTTP session consists of all of the HTTP requests from a single browser session. For example, if a user accesses an application and remains connected to the application for 10 minutes before exiting the application, all of the requests that the client makes in that 10 minute period are part of one session. If the user subsequently accesses the application, a new session is created. For more information on sessions, see Sessions in *Application development*.

The session manager allows to configure several settings for session tracking. These settings change how your application server treats HTTP sessions.
- Use the administrative console to administer session tracking (page 23)
- Use wsadmin to administer session tracking (page 24)

Use the administrative console to administer session tracking

To configure session tracking with the administrative console, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. Expand **Servers** and click **Application Servers**.

3. Click the name of your application server.
4. On the application server's page, click **Web Container**.
5. On the Web Container page, click **Session Management**.
6. Configure session management settings.
7. Click **Apply** or **OK**.
8. "Save the application server configuration" on page 84.

Use wsadmin to administer session tracking

To configure session tracking with wsadmin, follow these steps:
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the application server and assign it to the server variable:
   ```
   set server [$AdminConfig getid /Cell:myCell/Node:myNode/Server:myAppSvr/]
   ```
   where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, and *myAppSvr* is the name of your application server.
5. Identify the session manager for the application server and assign it to the smgr variable:
   ```
   set smgr [$AdminConfig list SessionManager $server]
   ```
6. Modify the attributes for session tracking.
   - To enable cookies and modify cookie settings, run this command:
     ```
     $AdminConfig modify $smgr {{enableCookies true}
      {defaultCookieSettings {{maximumAge 10}}}}
     ```
   - To enable protocol switch rewriting, run this command:
     ```
     $AdminConfig modify $smgr {{enableProtocolSwitchRewriting true}
      {enableUrlRewriting false} {enableSSLTracking false}}
     ```
   - To enable URL rewriting, run this command:
     ```
     $AdminConfig modify $smgr {{enableUrlRewriting true}
      {enableProtocolSwitchRewriting false} {enableSSLTracking false}}
     ```
   - To enable SSL tracking, run this command:
     ```
     $AdminConfig modify $smgr {{enableSSLTracking true}
      {enableProtocolSwitchRewriting false} {enableUrlRewriting false}}
     ```

   Note: These commands have been wrapped for display purposes.
7. Run this command to save your changes:
   ```
   $AdminConfig save
   ```

Manage substitution variables with the administrative console

In WebSphere Application Server - Express, you can use substitution variables to represent values in configuration files. For example, the variable LOG_ROOT represents the value /QIBM/UserData/WebASE/ASE5/*instance*/logs, where *instance* is the name of your instance. To change the location of the log files for your instance, change the value of the LOG_ROOT variable to the new path.

You can define variables for the cell, node, and server scopes. A variable applies to all configuration files in the scope for which it is defined. For example, WebSphere Application Server - Express defines the APP_INSTALL_ROOT variable for the node scope. The value defined for the node applies to all application servers within that node. If variables in multiple scopes have the same name, the variable with the narrowest scope takes precedence.

For instructions on adminstering substitution variables, see these sections:

- Create substitution variables (page 25)
- Modify substitution variables (page 25)
- Remove substitution variables (page 25)

Create substitution variables

To add a new substitution variable, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Manage WebSphere Variables**.
3. On the **WebSphere Variables** page, specify the scope for which you want to define the variable and click **Apply**.
4. Click **New**.
5. Specify a name and value for the variable.
6. Click **OK**.
7. "Save the application server configuration" on page 84.

Modify substitution variables

To modify a substitution variable, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Manage WebSphere Variables**.
3. On the **WebSphere Variables** page, specify the scope that contains the variable that you want to modify and click **Apply**.
4. Click the name of the variable that you want to modify.
5. Make your changes.
6. Click **OK**.
7. "Save the application server configuration" on page 84.

Remove substitution variables

To remove a substitution variable, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Manage WebSphere Variables**.
3. On the **WebSphere Variables** page, specify the scope that contains the variable that you want to remove and click **Apply**.
4. Select the checkbox for the variable that you want to remove.
5. Click **Delete**.
6. "Save the application server configuration" on page 84.

You can also use wsadmin to manage substitution variables for your appliation server instance. See "Use wsadmin to manage substitution variables" for information.

Use wsadmin to manage substitution variables

You can use wsadmin to configure a variable map for your application server. You can configure multiple variable maps, but they are all stored in the variables.xml file.

To configure a variable map for your application server, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains wsadmin:

```
cd /QIBM/ProdData/WebASE/ASE5/bin
```

3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the server and assign it to the server variable:

   ```
   set server [$AdminConfig getid /Cell:myCell/Node:myNode/Server:myAppSvr/]
   ```

   where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, and *myAppSvr* is the name of your application server.
5. Create an empty variable map or identify an existing variable map for the application server. Assign the variable map to the varMap variable.
   - To create an empty variable map for the server and assign it to the varMap variable, run this command:

     ```
     set varMap [$AdminConfig create VariableMap $server {}]
     ```

   - If your server has an existing variable map, run this command to get its configuration object and assign it to the varMap variable:

     ```
     set varMap [$AdminConfig getid /Cell:myCell/Node:myNode/Server:myAppSvr/VariableMap:/]
     ```

     where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, and *myAppSvr* is the name of your application server.
6. Set the variable map entry attributes. In this example, you create variable map entries APPLICATION_SPECIFIC_ROOT and APPLICATION_SPECIFIC_LIB_DIR.

   ```
   set nameattr1 [list symbolicName APPLICATION_SPECIFIC_ROOT]
     set valattr1 [list value "application/specific/root"]
     set nameattr2 [list symbolicName APPLICATION_SPECIFIC_LIB_DIR]
     set valattr2 [list value "/${APPLICATION_SPECIFIC_ROOT}/lib"]
     set attr1 [list $nameattr1 $valattr1]
     set attr2 [list $nameattr2 $valattr2]
     set attrs [list $attr1 $attr2]
   ```

   These commands return this output:

   ```
   {{symbolicName APPLICATION_SPECIFIC_ROOT} {value application/specific/root}}
    {{symbolicName APPLICATION_SPECIFIC_LIB_DIR} {value {${APPLICATION_SPECIFIC_ROOT}/lib}}}
   ```

7. To add the two new entries, run this command to modify the entries attribute in the variable map:

   ```
   $AdminConfig modify $varMap [subst {{entries {$attrs}}}]
   ```

8. Run this command to save your changes:

   ```
   $AdminConfig save
   ```

9. If you want to view the variable map, run this command:

   ```
   $AdminConfig showall $varMap
   ```

   The command returns this output:

   ```
   {entries {{{symbolicName APPLICATION_SPECIFIC_ROOT}
    {value application/specific/root}} {{symbolicName APPLICATION_SPECIFIC_LIB_DIR}
    {value ${APPLICATION_SPECIFIC_ROOT}/lib}}}}
   ```

Manage virtual hosts for your application server

A virtual host enables a single host machine to act as multiple hosts, allowing an administrator to independently manage resources on one physical machine. The Manage Virtual Hosts form allows you to view a list of defined virtual hosts, create a new virtual host, modify the properties of an existing virtual host, or remove a virtual host from your application server. For more information on virtual hosts, see "Virtual hosts" on page 28

To manage your virtual hosts on an application server, follow these steps:

1. Start the HTTP Server Administration interface.
2. Click the **Manage** tab.
3. Select an application server from the **Server** list.

4. Expand **Resource Configuration**.
5. Click **Manage Virtual Hosts**.

From the Manage Virtual Hosts form you can do the following:
- Create a new virtual host (page 27)
- Edit the host alias properties of existing virtual hosts (page 27)
- Remove a virtual host (page 28)

Create a new virtual host

Note: With the exception of step 1, these steps also apply to the Create Virtual Host wizard.
1. Click **Create**.
2. Specify the name of your virtual host in the **Virtual host name** field.
3. Click **Next**.
4. Select one these options.
   - **Create a new HTTP Server (powered by Apache)**
     Select this option to create a new HTTP Server (powered by Apache) instance.
     a. Click **Next**.
     b. Complete the Create a new HTTP Server (powered by Apache) form.
     c. Click **Next**. Continue to step 5.
   - **Select an existing HTTP Server (powered by Apache)**
     Select this option to use an existing HTTP Server (powered by Apache) instance.
     a. Click **Next**.
     b. Select one or more existing HTTP Servers (powered by Apache) from the list.
     c. Click **Next**. Continue to step 5.
   - **Select an existing Domino HTTP server**
     Select this option to use an existing Domino HTTP server.
     a. Click **Next**.
     b. Select one or more existing Domino HTTP servers from the list.
     c. Click **Next**. Continue to step 5.
5. Select the IP access types.
   - **Any host name or IP address**
     The wizard creates an IP access type from any available host name or valid IP address based on the HTTP Server (powered by Apache) or Domino HTTP server you selected. For example, http://*myserver:1023/*
   - **Specific access types**
     Select one or more access types.
     – IP address - The IP access type is a valid IP address. For example, http://*100.99.98.1:1023/*
     – Host short name - The IP access type is a valid host short name. For example, http://*myserver*:*1023/*
     – Domain qualified host name - The IP access type is a valid domain qualified host name. For example, http://*myserver.mycompany.com:1023/*
6. Click **Next**.
7. The summary page lists all of the choices you have made in the wizard. If any of the information displayed is incorrect, click **Back** until you reach the wizard form with the incorrect information and make your corrections. Click **Finish** to complete the wizard.

Edit the host alias properties of existing virtual hosts

1. Select a virtual host from the **Virtual hosts** table.

2. Click **Properties**.
3. You can edit one or more of the properties shown for the virtual host.
   - **Add a host alias setting**
     To add an alias setting, follow these steps:
     a. Click **Add.**
     b. Specify an IP address or hostname.
     c. Specify a port number.
     d. Click **Continue**.
   - **Change the properties of a host alias setting**
     To change the properties of an alias setting, follow these steps:
     a. Select a host alias setting from the table.
     b. Specify a new IP address and hostname.
     c. Specify a new port number.
     d. Click **Continue**.
   - **Remove a host alias setting**
     To remove an alias setting, follow these steps:
     a. Select an alias from the table.
     b. Click **Remove**.
     c. Click **Continue**.
4. Click **Apply** or **OK** to save your changes.

Remove a virtual host
1. Select a virtual host from the **Virtual hosts** table.
2. Click **Remove**.
3. Click **OK**.

For information on other ways to manage virtual hosts, see these topics:

"Administer virtual hosts with the WebSphere administrative console" on page 29
This topic describes how to use the administrative console to configure and manage virtual hosts for your application server.

"Use wsadmin to administer virtual hosts" on page 31
This topic describes how to use wsadmin to administer virtual hosts for your application server.

Virtual hosts

A virtual host is a configuration entity that allows WebSphere Application Server to treat multiple host machines or port numbers as a single logical host for configuration purposes. Each virtual host can be associated with multiple aliases. Each alias is a particular host name and port number. By combining multiple host machines into a single virtual host or by assigning host machines to different virtual hosts, you can separate and control which WebSphere Application Server resources are available for client requests.

When you configure WebSphere Application Server, you can associate a virtual host to one or more Web Modules. Each Web Module can be associated with one and only one virtual host. Each virtual host represents a virtual configuration that can have one or more Domain Name System (DNS) aliases. A DNS alias consists of a TCP/IP host name and port number used to request the servlet (for example, yourhost:8000). A virtual host can be associated with any number of DNS aliases.

A client request for a servlet, JavaServer Pages (JSP) 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, JSP 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 a matching virtual host group or URI group is not found, an error is returned to the browser. 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.

The first time that you start the WebSphere Application Server application server, a default virtual host (named default_host) is configured. The DNS aliases for the default virtual host are configured as *:80 and *:9080, where port 80 is the HTTP server port and port 9080 is the port for the default server's servlet engine. Because the transport type used to communicate with your application server is HTTP, WebSphere Application Server is capable of accepting requests directly from the browser.

Use the administrative console to add or change DNS aliases if you want to use ports other than the default ports listed above. If you do make a change to the DNS aliases, also regenerate the plug-in configuration, which can also be done from the administrative console. These are some situations in which you need to make additions or changes:
- 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.
- If you want to make HTTPS requests, which use Secure Sockets Layer (SSL), add port 443 to each of the aliases. Port 443 is the default port for SSL requests. If your Web server (HTTP server) instance is listening for SSL requests on a port other than 443, add that port number to each of the aliases.
- You want to use a port for the application server other then default port (9080).
- You want to use other aliases that are not listed.

The virtual host also maintains a list of Multipurpose Internet Mail Extensions (MIME) types that it processes. MIME is an Internet standard for multimedia e-mail, including graphics, audio, and fax.

Administer virtual hosts with the WebSphere administrative console

A virtual host is a configuration entity that allows WebSphere Application Server - Express to treat multiple host names, IP addresses, or port numbers as a single logical host. For more information, see "Virtual hosts" on page 28.

You can use the WebSphere administrative console to perform these tasks with virtual hosts:
- Configure new virtual hosts (page 29)
- Modify virtual hosts (page 30)
- Remove virtual hosts (page 30)

Configure new virtual hosts

To configure a new virtual host, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Virtual Hosts**.
3. On the **Virtual Hosts** page, click **New**.
4. Specify a name for the virtual host.
5. Click **Apply**.
6. Configure one or more aliases for the virtual host.
   a. On the virtual host's detail page, click **Host Aliases**.
   b. On the **Host Aliases** page, click **New**.
   c. Specify a host name and port number for the alias.
   d. Click **OK**.

7. Configure MIME type extension mappings defined the virtual host. By default, several common mappings are configured for the new virtual host. To add a mapping, follow these steps:
   a. On the virtual host's detail page, click **MIME Types**.
   b. On the **MIME Types** page, click **New**.
   c. Specify a MIME type and extensions that map to it.
   d. Click **OK**.
8. "Save the application server configuration" on page 84.
9. "Regenerate the Web server plugin configuration" on page 41.

Modify virtual hosts

To modify a virual host, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Virtual Hosts**.
3. On the **Virtual Hosts** page, click the name of the virtual host that you want to modify.
4. Make your changes. You can add, modify, and remove host aliases and MIME type mappings. (To add host aliases and MIME types, see the steps above.)
   - To modify a host alias, follow these steps:
     a. Click **Host Aliases**.
     b. On the **Host Aliases** page, click the name of the alias that you want to modify.
     c. Make your changes.
     d. Click **OK**.
   - To remove a host alias, follow these steps:
     a. Click **Host Aliases**.
     b. On the **Host Aliases** page, select the checkbox for the alias that you want to remove.
     c. Click **Delete**.
   - To modify the MIME type mappings, follow these steps:
     a. Click **MIME Types**.
     b. On the **MIME Types** page, click the MIME type that you want to modify.
     c. Make your changes.
     d. Click **OK**.
   - To remove a MIME type, follow these steps:
     a. Click **MIME Types**.
     b. On the **MIME Types** page, select the checkbox for the MIME type that you want to remove.
     c. Click **Delete**.
5. "Save the application server configuration" on page 84.
6. "Regenerate the Web server plugin configuration" on page 41.

Remove virtual hosts

To remove a virtual host, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Virtual Hosts**.
3. On the **Virtual Hosts** page, select the checkbox for the virtual host that you want to remove.
4. Click **Delete**.
5. "Save the application server configuration" on page 84.
6. "Regenerate the Web server plugin configuration" on page 41.

Use wsadmin to administer virtual hosts

A virtual host is a configuration entity that allows WebSphere Application Server - Express to treat multiple host names, IP addresses, or port numbers as a single logical host. For more information, see "Virtual hosts" on page 28.

To configure virtual hosts with wsadmin, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the cell and assign it to the cell variable:
   ```
   set cell [$AdminConfig getid /Cell:myCell]
   ```
   where *myCell* is the name of your application server cell.
5. Run this command to to create a new virtual host:
   ```
   $AdminConfig create VirtualHost $cell {{name newVHost}}
   ```
   where *newVHost* is the name of the new virtual host.
6. Run this command to save your changes:
   ```
   $AdminConfig save
   ```

Configure database access for your application

If your application needs to access a backend database, you must configure JDBC resources in your application server. Your application uses JDBC providers to connect to the database. For information on configuring database access for your application, see these topics.

"Manage JDBC providers for your application server"
JDBC providers represent JDBC drivers, which your applications use to access databases. This topic describes how to configure and manage JDBC providers for your application server.

"Manage data sources for your application server" on page 34
Data sources represent the databases that your applications use. This topic describes how to configure and manage data sources for your application server.

"Administer JDBC providers and data sources with the administrative console" on page 36
This topic describes how to use the administrative console to configure and manage JDBC providers and data sources.

"Use wsadmin to configure a JDBC provider" on page 37
This topic describes how to configure a JDBC provider with wsadmin.

"Use wsadmin to configure a new data source" on page 38
This topic describes how to create a new data source with wsadmin.

"Configure connection pooling" on page 38
You can create a pool of database connections that your applications use to access data sources. This topic describes how to use the administrative console and wsadmin to configure connection pooling for your application server.

Manage JDBC providers for your application server

A JDBC provider specifies how database connections are created for data sources that are associated with the provider and which JDBC driver is used for the connection. Creating JDBC providers define the JDBC driver that your application uses. The Manage JDBC Providers form allows you to view the list of defined JDBC providers, create a new JDBC provider, modify the properties of an existing JDBC provider, or remove a JDBC provider from your application server. WebSphere Application Server - Express for iSeries supports these JDBC drivers:

- DB2 UDB for iSeries (IBM Developer Kit for Java JDBC driver - V5R2 and later)
- DB2 UDB for iSeries (IBM Developer Kit for Java JDBC driver, JTA-enabled - V5R2 and later)
- DB2 UDB for iSeries (IBM Developer Kit for Java JDBC driver - V5R1 and earlier)
- DB2 UDB for iSeries (IBM Developer Kit for Java JDBC driver, JTA-enabled - V5R1 and earlier)
- DB2 UDB for iSeries (IBM Toolbox for Java JDBC driver)
- DB2 UDB for iSeries (IBM Toolbox for Java JDBC driver, JTA-enabled)

For additional information on the IBM Toolbox for Java JDBC driver, see "The IBM Toolbox for Java JDBC driver" on page 33.

To manage your JDBC providers on an application server, follow these steps:

1. Start the HTTP Server Administration interface.
2. Click the **Manage** tab.
3. Select an application server from the **Server** list.
4. Expand **Resource Configuration**.
5. Click **Manage JDBC Providers**.

From the Manage JDBC Providers form, you can do the following:

- Create a new JDBC provider (page 32)
- Edit the properties of existing JDBC provider (page 33)
- Remove a JDBC provider (page 33)

Create a new JDBC provider

Note: With the exception of step 1, these steps also apply to the Create JDBC Provider wizard.

1. Click **Create**.
2. Specify a name for your new JDBC provider in the **JDBC provider name** field.
3. Click **Next.**
4. Select the database access method.
   - **Local - IBM Developer Kit for Java JDBC(TM) driver**
     Select this option if the application server and the database are located on the same system.
   - **Remote - IBM Toolkit for Java JDBC(TM) driver**
     Select this option if the application server and the database are located on separate systems or logical partitions (LPAR).
5. Select Java Transaction API (JTA) enablement.
   - **Enabled**
     JTA is enabled. JTA supports two-phase commit, which is a database protocol that ensures a transaction is completely committed to the database. This is necessary for transactions that are comprised of multiple updates to one or more databases. If one of the updates fails, the entire transaction fails, and all updates that were made in the transaction are rolled back. This ensures that only complete information is saved.
   - **Disabled**
     JTA is disabled. If you do not need to use two-phase commit, disable JTA. For example, if a transaction only updates one database, JTA is not required. Disabling JTA can improve application performance.

6. The summary page lists all of the choices you have made in the wizard. If any of the information displayed is incorrect, click **Back** until you reach the wizard form with the incorrect information and make your corrections. Click **Finish** to complete the wizard.

Edit the properties of existing JDBC provider

1. Select a JDBC provider from the **JDBC providers** table.
2. Click **Properties**.
3. You can edit one or more properties shown for the JDBC provider.
   - **Edit the JDBC provider name or description**
     a. Click the **General** tab in the JDBC Provider Properties form. This value allows for an optional user-defined descriptions of the JDBC provider.
     b. Specify a new name for the JDBC provider in the **JDBC provider name** field.
     c. Specify a new description for the JDBC provider in the **JDBC provider description** field.
     d. Specify a new JDBC class path in the **Class path** field. This field is only available if the database access method is remote.
   - **Add, edit, or delete a data source**
     a. Click the **Data Sources** tab in the JDBC Provider Properties form. For more information on managing data sources, see "Manage data sources for your application server" on page 34.
4. Click **Apply** or **OK** to save your changes.

Remove a JDBC provider

1. Select a JDBC provider from the table.
2. Click **Remove**.
3. Click **OK**

The IBM Toolbox for Java JDBC driver: WebSphere Application Server - Express supports the IBM Toolbox for Java JDBC driver. This JDBC driver is included with the IBM Toolbox for Java product.

IBM Toolbox for Java is a library of Java classes that are optimized for accessing iSeries and AS/400 data and resources. You can use the IBM Toolbox for Java JDBC driver to access local or remote DB2 UDB for iSeries 400 databases from server-side and client-side Java applications that run on any platform that supports Java.

IBM Toolbox for Java is available in these versions:

- **IBM Toolbox for Java licensed program**
  The licensed program is available with every i5/OS release, Version 4 Release 2 (V4R2) or later. You can install the licensed program on your iSeries system, and then either copy the IBM Toolbox for Java JAR file (jt400.jar) to your system or update your system classpath to locate the server installation. For product documentation for IBM Toolbox for Java, see 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 the IBM Toolbox for Java: Downloads page.

  You can also download the JTOpen Programming Guide. The guide includes instructions for installing JTOpen, as well as information about the JDBC driver.

The JDBC driver for both versions supports JDBC 2.0. For more information about IBM Toolbox for Java and JTOpen, see the product Web site at http://www-1.ibm.com/servers/eserver/iseries/toolbox/index.html.

If you are running WebSphere Application Server - Express, Version 5, on a non-iSeries platform, download the JTOpen version of IBM Toolbox for Java.

To configure the jt400.jar file, do the following:

1. Download the jt400.jar file. Save the JAR file to a directory, such as /JDBCDrivers/Toolbox.
2. "Start the WebSphere administrative console" on page 83.
3. In the administrative console topology tree, expand **Environment** and click **Manage WebSphere Variables**.
4. Verify that the scope is set to **Node**.
5. Click **OS400_TOOLBOX_JDBC_DRIVER_PATH**.
6. In the **Value** field, specify the fully qualified path of the directory that contains the jt400.jar file. Do not include the jt400.jar file in this value. For example, if the file is saved in /JDBCDrivers/Toolbox, the value of the variable is /JDBCDrivers/Toolbox. This path is referenced in the jdbc-resource-provider-templates.xml file, which is used to configure JDBC resource providers.
7. "Save the application server configuration" on page 84.

To use a Toolbox JDBC driver, select **UDB DB2 for iSeries (Toolbox)** or **UDB DB2 for iSeries (Toolbox XA)** when you create a new JDBC provider. When the provider is created, the classpath field contains this value: ${OS400_TOOLBOX_JDBC_DRIVER_PATH}/jt400.jar.

Manage data sources for your application server

A data source represents a specific database that is used by one or more applications installed in your application server. Data sources provide connection pooling and a JNDI name which can be used by applications to access the data source. The Manage Data Sources form allows you to view a list of all the defined data sources for an application server, create new data sources, modify the properties of an existing data source, or remove a data source.

To manage your data sources for an application server, follow these steps:

1. Start the HTTP Server Administration interface.
2. Click the **Manage** tab.
3. Select an application server from the **Server** list.
4. Expand **Resource Configuration**.
5. Click **Manage Data Sources**.

From the Manage Data Sources form, you can do the following:

- Create a new data source (page 34)
- Edit the properties of an existing data source (page 35)
- Remove a data source (page 36)

Create a new data source

Note: With the exception of step 1, the below steps can be used with the Create Data Source wizard.

1. Click **Create**.
2. Specify a name for the data source in the **Data source name** field. This value specifies the user-defined name of the data source.
3. Click **Next**.
4. Select a JDBC provider for your data source.

- **Create new JDBC provider**
  Select this option to create a new JDBC provider for your data source.
  a. Click **Next**.
  b. See Create a new JDBC provider (page 32) for instructions on how to create a JDBC provider for your data source.
- **Select an existing JDBC provider**
  Select this option to use a preexisting JDBC provider.
  a. Click **Next**.
  b. Select a JDBC provider from the table.
  c. Click **Next**.

5. Specify a name for the database in the **Database name** field. This value specifies the name of the database that the current data source represents. If the database resides on the same physical machine as the application server, this value is *LOCAL. If the database is on a different machine, this value is the name of the server where the database resides.
6. Specify a name for the collection, schema or library for the database in the **Collection, Schema or Library name** field. This value specifies additional information to access the database data. Collection, schema, and library all represent organized collections of named objects (tables) in a database. When you specify the collection name, you do not need to use fully qualified table names in the application code.
7. Click **Next**.
8. Specify a name for the JNDI in the **JNDI name** field or keep the provided value. The wizard provides a JNDI name for the data source. It is recommended that you use this default value. The Java Naming and Directory Interface (JNDI) provides naming and directory functionality for Java applications and resources. Applications use JNDI to access named Java objects. The conventional format of JNDI names for data sources is jdbc/*dataSourceName*.
9. Click **Next**.
10. The summary page lists all of the choices you have made in the wizard. If any of the information displayed is incorrect, click **Back** until you reach the wizard form with the incorrect information and make your corrections. Click **Finish** to complete the wizard.

Edit the properties of an existing data source

1. Select a data source from the **All data sources** table.
2. Click **Properties**.
3. You can edit one or more of the properties shown for the data source.
   - **Data source name**
     Specify a new name in the **Data source name** field. This value specifies the user-defined name of the data source.
   - **Database name**
     Specify a name for the database in the **Database name** field. This value specifies the name of the database that the current data source represents. If the database resides on the same physical machine as the application server, this value is *LOCAL. If the database is on a different machine, this value is the name of the server where the database resides.
   - **Collection, schema, or library name**
     Specify a new collection, schema, or library name. Collections, schemas, and libraries all represent organized collections of named objects (tables) in a database.
   - **JNDI name**
     Specify a name for the JNDI in the **JNDI name** field or keep the provided value. The Java Naming and Directory Interface (JNDI) provides naming and directory functionality for Java applications and resources. Applications use JNDI to access named Java objects. The conventional format of JNDI names for data sources is jdbc/*dataSourceName*.

- **Data source description**
  Specify a brief description in the **Data source description** field. This value specifies the user-defined description of the data source.

4. Click **Apply** or **OK** to save your changes.

Remove a data source

1. Select a data source from the table.
2. Click **Remove**.
3. Click **OK**.

Administer JDBC providers and data sources with the administrative console

These sections describe how use the administrative console to configure, modify, and remove JDBC providers and data sources:

- Configure JDBC providers and data sources (page 36)
- Modify JDBC providers and data sources (page 36)
- Remove JDBC providers and data sources (page 37)

Configure JDBC providers and data sources

To configure a JDBC provider, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Resources** and click **JDBC Providers**.
3. Verify that the scope is set to the server level.
4. On the **JDBC Providers** page, click **New**.
5. In the **JDBC Providers** field, select one of the supported providers. WebSphere Application Server - Express for iSeries supports all of the DB2 UDB for iSeries drivers. For the IBM Developer Kit for Java JDBC drivers, select the correct driver for your version of i5/OS.
6. Click **Apply**.
7. The name and implementation classname for the provider are provided for you.

   Note: The **Native Library Path** is not configurable on iSeries.
8. Click **Apply** or **OK**.

After you create a JDBC provider, configure data sources for the provider. To configure a data source, follow these steps:

1. On the **JDBC Providers** page, click the name of the JDBC provider for which you want to configure a data source.
2. On the JDBC provider's detail page, click **Data Sources**.
3. On the **Data Sources** page, click **New**.
4. Specify a name for the data source. You can also specify optional properties on this page.
5. Click **Apply**.
6. You can configure these additional properties for the data source:
   - **Connection Pooling**
   - **Custom Properties**
7. "Save the application server configuration" on page 84.

Modify JDBC providers and data sources

To modify a JDBC provider, follow these steps:

1. "Start the WebSphere administrative console" on page 83.

2. In the topology tree, expand **Resources** and click **JDBC Providers**.
3. Verify that the scope is set to the server level.
4. On the **JDBC Providers** page, click the name of the provider that you want to modify.
5. Make your changes.
6. Click **Apply**.
7. "Save the application server configuration" on page 84.

You can also modify data sources for the provider. To modify a data source, follow these steps:
1. On the **JDBC Providers** page, click the name of the JDBC provider that includes the data source that you want to modify.
2. On the JDBC provider's detail page, click **Data Sources**.
3. On the **Data Sources** page, click the name of the data source that you want to modify.
4. Make your changes.
5. Click **Apply**.
6. To modify additional properties for the data source, click **Connection Pooling** or **Custom Properties** at the bottom of the page.
7. "Save the application server configuration" on page 84.

Remove JDBC providers and data sources

To remove a JDBC provider follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Resources** and click **JDBC Providers**.
3. Verify that the scope is set to the server level.
4. On the **JDBC Providers** page, select the checkbox for the provider that you want to remove.
5. Click **Delete**.
6. "Save the application server configuration" on page 84.

To remove a data source, follow these steps:
1. On the **JDBC Providers** page, click the name of the JDBC provider that contains the data source that you want to remove.
2. On the JDBC provider's detail page, click **Data Sources**.
3. On the **Data Sources** page, select the checkbox for the data source you want to remove.
4. Click **Delete**.
5. "Save the application server configuration" on page 84.

Use wsadmin to configure a JDBC provider

To configure a JDBC provider with wsadmin, follow these steps:
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the parent ID and assign it to the node variable:
   ```
   set node [$AdminConfig  getid  /Cell:myCell/Node:myNode/Server:myAppSvr/]
   ```
   where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, and *myAppSvr* is the name of your application server.
5. Run this command to identify the required attributes:

```
$AdminConfig required JDBCProvider
```

The command displays this output:

```
Attribute                 Type
name                      String
implementationClassName   String
```

6. Run these commands to set the required attributes and assign them to the jdbcAttrs variable:

   ```
   set n1 [list name JDBC1]
   set implCN [list implementationClassName myclass]
   set jdbcAttrs [list $n1 $implCN]
   ```

   You can modify the example to set optional attributes for JDBC provider. For a list of all of the attributes of a JDBC provider, run this command:

   ```
   $AdminConfig attributes JDBCProvider
   ```

7. Run this command to create the new JDBC provider:

   ```
   $AdminConfig create JDBCProvider $server $jdbcAttrs
   ```

8. Run this command to save your changes:

   ```
   $AdminConfig save
   ```

Use wsadmin to configure a new data source

To configure a data source with wsadmin, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:

   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```

3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the parent ID:

   ```
   set newjdbc [$AdminConfig getid /Cell:myCell/Node:myNode/
   Server:myAppSvr/JDBCProvider:JDBC1/]
   ```

   where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, *myAppSvr* is the name of your application server, and *JDBC1* is the name of your JDBC provider.

   Note: This command has been wrapped for display purposes.

5. Run this command to display the required attributes for a data source:

   ```
   $AdminConfig required DataSource
   ```

   The command displays this output:

   ```
   Attribute        Type
   name             String
   ```

6. Run these commands to set the required attributes:

   ```
   set name [list name DS1]
   set dsAttrs [list $name]
   ```

   where *DS1* is the name of the data source that you want to create.

7. Run this command to create a data source:

   ```
   set newds [$AdminConfig create DataSource $newjdbc $dsAttrs]
   ```

8. Run this command to save your changes:

   ```
   $AdminConfig save
   ```

Configure connection pooling

Connection pooling creates a pool of database connections that your applications can use to connect to the database, and can reduce the amount of resources dedicated to establishing and maintaining database connections. Because Web applications typically make more connections to a database than other applications, connection pooling is well suited to Web applications.

- Use the administrative console to configure connection pooling (page 39)
- Use wsadmin to configure connection pooling (page 39)

Use the administrative console to configure connection pooling

1. "Start the WebSphere administrative console" on page 83.
2. Expand **Resources** and click **JDBC Providers**.
3. Click the name of the JDBC provider that is associated with the data source for which you want to configure connection pooling.
4. Click **Data Sources**.
5. On the Data Sources page, click the name of the data source for which you want to configure connection pooling.
6. Click **Connection Pool**.
7. On the Connection Pools page, configure connection pooling settings.
8. Click **Apply** or **OK**.
9. "Save the application server configuration" on page 84.

Use wsadmin to configure connection pooling

To configure connection pooling with wsadmin, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the parent ID:
   ```
   set newds [$AdminConfig getid
    /Cell:myCell/Node:myNode/Server:myAppSvr/
    JDBCProvider:JDBC1/DataSource:DS1/]
   ```
   where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, *myAppSvr* is the name of your application server, *JDBC1* is the name of your JDBC provider, and *DS1* is the name of your the data source for which you want to create a connection pool.

   Note: This command has been wrapped for display purposes.
5. Run this command to create a new connection pool:
   ```
   $AdminConfig create ConnectionPool $newds {}
   ```
6. Run this command to save your changes:
   ```
   $AdminConfig save
   ```

Administer mail resources

JavaMail provides general mail facilities for reading and sending mail. WebSphere Application Server - Express supplies a built-in mail provider that supports three protocol providers: Simple Mail Transfer Protocol (SMTP), Internet Message Access Protocol (IMAP) and Post Office Protocol (POP3). For more information on JavaMail and mail providers, see JavaMail.

You can use the administrative console and wsadmin to administer mail resources.

- Administer mail resources with the administrative console (page 39)
- Use wsadmin to configure a new mail provider (page 40)

Administer mail resources with the administrative console

For information on configuring, modifying, and removing mail providers, see these sections:

- Configure mail providers (page 40)
- Modify mail providers (page 40)
- Remove mail providers (page 40)

Configure mail providers

To configure a new mail provider, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Resources** and click **Mail Providers**.
3. On the **Mail Providers** page, click **New**.
4. Specify a name for the mail provider.
5. Click **Apply**.
6. You can configure these additional properties for the mail provider:
   - **Protocol Providers**
   - **Mail Sessions**
7. "Save the application server configuration" on page 84.

Modify mail resources

To modify a mail provider, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Resources** and click **Mail Providers**.
3. On the **Mail Providers** page, click the name of the provider that you want to modify.
4. Make your changes.
5. Click **OK**.
6. "Save the application server configuration" on page 84.

Remove mail providers

To remove a mail provider, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Resources** and click **Mail Providers**.
3. On the **Mail Providers** page, select the checkbox for the name of the provider that you want to remove.
4. Click **Delete**.
5. "Save the application server configuration" on page 84.

Use wsadmin to configure a new mail provider

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the parent ID:
   ```
   set server [$AdminConfig getid /Cell:myCell/Node:myNode/Server:myAppSvr/]
   ```
   where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, and *myAppSvr* is the name of your application server.
5. Run this command to get the required attributes for a mail provider:
   ```
   $AdminConfig required MailProvider
   ```
6. Run this command to set the required attributes:
   ```
   set name [list name mpname]
   set mpAttrs [list $name]
   ```
   where *mpname* is the name of the mail provider that you want to create.

7. Run this command to create the mail provider:
   ```
   set newmp [$AdminConfig create MailProvider $server $mpAttrs]
   ```
8. Run this command to save your changes:
   ```
   $AdminConfig save
   ```

Regenerate the Web server plugin configuration

The GenPluginCfg script regenerates the plugin-cfg.xml file, which stores runtime configuration information for the Web server plugin. The file is stored in the /QIBM/UserData/WebASE/ASE5/*instance*/config/cells directory, where *instance* is the name of your instance. For information on running Qshell scripts, see "Configure Qshell to run WebSphere Application Server - Express scripts" on page 92.

Regenerate the plugin configuration after you make changes to the application server. This list describes some, but not all, situations in which you should regenerate the plugin-cfg.xml file:
- After you install or remove an enterprise application.
- After you add or remove servlets and mappings from an application.
- After you change the configuration for the plugin, a virtual host, or a transport.

The application server must be running when you run this script.

Authority

To run this script, your user profile must have *ALLOBJ authority.

Usage
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the GenPluginCfg script:
   ```
   GenPluginCfg -instance instance
   ```
   where *instance* is the name of the application server instance for which you want to regenerate the plugin-cfg.xml file.

Syntax

The syntax of the GenPluginCfg script is:
```
GenPluginCfg -instance instance [ -config.root cfgdirectory ]
 [ -cell.name cellname ] [ -node.name nodename ]
[ -output.file.name filename ]
 [ -debug yes | no ]
```

Parameters

The parameters of the GenPluginCfg script are:
- **-instance**
  This is a required parameter. The value *instance* specifies the name of your application server instance.
- **-config.root**
  This is an optional parameter. The value *cfgdirectory* specifies the configuration directory that contains the plugin-cfg.xml file. The default value is /QIBM/UserData/WebASE/ASE5/*instance*/config/cells, where *instance* is the name of your instance.
- **-cell.name**
  This is an optional parameter. The value *cellname* specifies the name of the cell in which your

application server resides. The default is the value specified for the WAS_CELL environment variable found in the file /QIBM/UserData/WebASE/ASE5/*instance*/bin/setupCmdLine, where *instance* is the name of your instance. This value is case sensitive.

- **-node.name**
  This is an optional parameter. The value *nodename* specifies the name of the node in which your application server resides. The default is the value specified for the WAS_NODE environment variable found in the file /QIBM/UserData/WebASE/ASE5/*instance*/bin/setupCmdLine, where *instance* is the name of your instance. This value is case sensitive.
- **-output.file.name**
  This is an optional parameter. The value *filename* specifies the name of the file for the new plugin configuration. The default value is the plugin-cfg.xml file in the /QIBM/UserData/WebASE/ASE5/*instance*/config/cells/ directory, where *instance* is the name of your instance.
- **-debug**
  This is an optional parameter. If you specify yes, the script generates exception trace information. If the script throws exceptions, run it with this parameter to view the trace information. The default value is no.

Example

```
GenPluginCfg -instance myAppSvr
```

This example regenerates the plugin-cfg.xml file for the instance myAppSvr.

You can also regenerate the Web server plugin configuration with the administrative console and wsadmin. See "Regenerate the Web server plugin configuration with the console and wsadmin."

Regenerate the Web server plugin configuration with the console and wsadmin

When you make changes to an application server, you must regenerate the Web server plugin configuration file to ensure that the correct set of requests are forwarded to the WebSphere Application Server - Express runtime. To regenerate the plugin configuration, follow these steps:

- Use the administrative console to regenerate the Web server plugin configuration (page 42)
- Use wsadmin to regenerate the Web server plugin configuration (page 42)

Use the administrative console to regenerate the Web server plugin configuration

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Update Web Server Plugin**.
3. On the **Update web server plugin configuration** page, click **OK**.

Note: If you make manual updates to the plugin configuration file, those updates may be overwritten when you regenerate the file. To avoid overwriting your manual updates, make a backup copy of the plugin configuration file before your regenerate the file, then use its contents to manually update the configuration file after you regenerate it.

Use wsadmin to regenerate the Web server plugin configuration file

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the web server plugin configuraiton file generator MBean and assign it to the pluginGen variable:
   ```
   set pluginGen [$AdminControl completeObjectName type=PluginCfgGenerator,*]
   ```
5. Run this command to generate the updated plugin configuration file:

```
$AdminControl invoke $pluginGen generate "/QIBM/UserData/WebASE/ASE5/instance/
 /QIBM/UserData/WebASE/ASE5/instance/config myCell null null null"
```

where *instance* is the name of your application server instance and *myCell* is the name of the cell that contains your application server.

Note: This command has been wrapped for display purposes.

This example command invokes the generate operation on the MBean passing the instance directory, the configuration root directory, and the cell name.

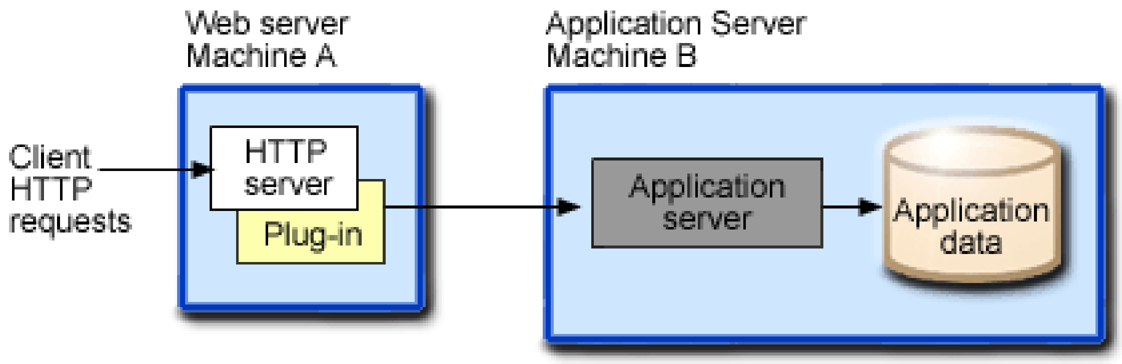
6. Run this command to save your changes:

```
$AdminConfig save
```

Configure a remote HTTP topology

In a remote HTTP topology, the application server and the HTTP server are hosted on separate machines or logical partitions. Your application server connects to the new HTTP server instance to receive client requests.

This figure shows an example of a remote HTTP topology:



In this sample topology, Machine A hosts the Web server and receives HTTP requests from clients. The Web server uses HTTP or HTTPS protocol to forward the requests to the application server on Machines B.

To create an HTTP server instance on a machine or logical partition that does not host your application server, use the crtpligininst script in Qshell. For information on running Qshell scripts, see "Configure Qshell to run WebSphere Application Server - Express scripts" on page 92.

Before you can run the script, you must install the *BASE option of WebSphere Application Server - Express (5722IWE *BASE) on the machine or logical partition that hosts your Web server instance. When you install the *BASE option, the QEJBSVR profile is created on the machine. The script also requires that one of these profiles is also on the machine:

- **QTMHHTTP** if you are using IBM HTTP Server for i5/OS as your Web server
- **QNOTES** if you are using Domino as your Web server

The crtpligininst script creates and configures the directory structures that the Web server plugins use to find a plugin-cfg.xml file generated by your application server instance. After the plugin-cfg.xml file has been generated on your application server system with the correct configuration information, you must move the file from the system that hosts your application server to the system that hosts your HTTP server instance. After you move the plugin-cfg.xml file to the Web server machine, you must run the crtpligininst script one more time to set up the appropriate access authorities for the plugin.

To point to the location of the HTTP server instance, the configuration file uses a host alias within the virtual host that you choose. For information on configuring a virtual host, see “Manage virtual hosts for your application server” on page 26.

Authority

To run this script, your iSeries user profile must have *ALLOBJ authority.

Usage

Run the crtplugininst script on the machine or logical partition that hosts your HTTP server instance.

1. On the CL command line, enter the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/bin
   ```
3. Run the crtplugininst script:
   ```
   crtplugininst -instance instance
   ```
   where *instance* is the name of your application server instance.
4. After the script runs, copy the plugin-cfg.xml file from your application server instance to the system that hosts your HTTP server instance. For example, for an application server named myAppSvr, the file is located in the /QIBM/UserData/WebASE/ASE5/myAppSvr/config/cells directory. Copy the file to the /QIBM/UserData/WebASE/ASE5/myAppSvr/config/cells directory on the machine or logical partition that hosts your Web server.
5. Run the crtplugininst script again. Specify the same value for the -instance parameter. When the script is finished, it displays this message:
   ```
   The Creation of a WebSphere Application Server - Express plugin instance completed successfully.
   ```

Note: Whenever you regenerate the Web server plugin configuration for your application server, you must copy the new plugin-cfg.xml file to the Web server machine. After you copy the new plugin-cfg.xml file to the Web server machine, you must run the crtplugininst script to ensure that the new file has the correct authorities.

Syntax

The syntax of the script is shown below.

```
crtplugininst -instance instance
```

where *instance* is the name of your WebSphere Application Server - Express application server instance.

The script creates a directory structure based on the value that you specify for the -instance parameter. For example, for an instance named myAppSvr, the crtplugininst creates these directories:

- /QIBM/UserData/WebASE/ASE5/myAppSvr
- /QIBM/UserData/WebASE/ASE5/myAppSvr/config
- /QIBM/UserData/WebASE/ASE5/myAppSvr/config/cells
- /QIBM/UserData/WebASE/ASE5/myAppSvr/logs
- /QIBM/UserData/WebASE/ASE5/myAppSvr/etc

Configure security settings for your application server

You can use the WebSphere administrative console to perform the tasks required to configure security for your application server and applications. For information on configuring security for WebSphere Application Server - Express, see Security.

These topics provide information on granting and revoking user authority to your application server instance:

“Grant authority to an instance”
This topic describes how to use the grtwasaut script in Qshell to grant authority to an application server instance.

“Revoke authority to an instance” on page 47
This topic describes how to use the rvkwasaut script in Qshell to revoke authority to an application server instance.

“Encode password data” on page 48
This topic describes how to use the EncAuthDataFile script to encode passwords in a text file.

“Encode passwords in properties files” on page 48
This topic describes how to use the PropFilePasswordEncoder script to encode passwords in a properties file.

Grant authority to an instance

The grtwasaut script grants a user authority to an instance and the objects associated with it.

Authority

To run this script, your user profile must have *ALLOBJ authority.

Usage

To grant authority to objects and directories in an instance, run the grtwasaut script from the Qshell command line. To run the script, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the grtwasaut script:
   ```
   grtwasaut -instance instance -user usrprf | -authlist authlist
    -dtaaut dataAuth -objaut objAuth
   ```
   where *instance* is the instance to which you are granting authority, *usrprf* is the user profile to which you are granting authority, *authlist* is the authorization list to which you are granting authority, *dataAuth* specifies the data authorities that you are granting to the user specified by the -user parameter and *objAuth* specifies the object authorities that you are granting to the user specified by the -user parameter. You do not need to specify both the -user and -authlist parameters, but you must specify at least one of them.

Syntax

The syntax of the script is:

```
grtwasaut -instance instance { -user usrprf | -authlist authlist }
 { -dtaaut dataAuth | -objaut objectAuth } [ -object path ]
 [ -recursive ] [ -verbose ] [ -help ]
```

Note: When you run the grtwasaut script, you must specify these paramters:

- -user, -authlist, or both
- -dtaaut, objaut, or both

Parameters

The parameters of the script are:

- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance to which you are granting authority.
- **-user**
  The value *usrprf* specifies the i5/OS user profile to which you are granting authority. To grant authority to multiple user profiles, specify all of the user profiles with a single -user parameter. Enclose the list of profiles in double quotation marks ("). For example, to grant authority to usrprf1 and usrprf2, specify `-user "usrprf1 usrprf2"`. You must specify -user, -authlist, or both.
- **-authlist**
  The value *authlist* specifies the i5/OS authorization list to which you are granting authority. You must specify -user, -authlist, or both.
- **-dtaaut**
  The value *dataAuth* specifies the data authorities that you are granting to the user specified by the -user parameter. Valid values are none, rwx, rx, rw, wx, r, w, x, exclude, autl, and same. The specified value replaces the user's current data authorities to the object. You must specify -dtaaut, -objaut, or both. For more information on the values for this parameter, see the CHGAUT (Change Authority) command description.
- **-objaut**
  The value *objAuth* specifies the object authorities that you are granting to the user specified in the -user parameter. Valid values are none, all, objexist, objmgt, objalter, objref, and same. The specified value replaces the user's current object authorities to the object. You must specify -dtaaut, -objaut, or both. For more information on the values for this parameter, see the CHGAUT (Change Authority) command description.
- **-object**
  The value *path* specifies the subdirectory or partially qualified object name to which you are granting authority. The instance root is prepended to the value to get the fully-qualified path. If you do not specify this parameter, the default value is the instance root. To grant authority to multiple objects, you must run the script for each object.
- **-recursive**
  This optional parameter specifies whether to grant authority to all subdirectories. If you do not specify this parameter, authority is granted only to the object specified with the -object parameter, or the instance root directory if the -object parameter is not specified. This parameter applies to all objects specified with -object parameters.
- **-verbose**
  This optional parameter turns on verbose messages, which can be helpful if you need to debug the script.
- **-help**
  This optional parameter displays the help message. If you specify this parameter, the script ignores all other parameters.

Examples

In this example, the user profiles johndoe and jsmith are granted rwx authority to the instance devinst and the associated objects.

```
grtwasaut -instance devinst -user "johndoe jsmith" -dtaaut rwx -recursive
```

In this example, the user profiles johndoe and jsmith are granted rwx authority to the installedApps subdirectory and all nested objects in the installedApps subdirectory.

```
grtwasaut -instance devinst -object installedApps -user "johndoe jsmith" -dtaaut rwx -recursive
```

Revoke authority to an instance

You can use the rvkwasaut script in Qshell to revoke authority to objects and directories in an instance.

Authority

To run this script, your user profile must have *ALLOBJ authority.

Usage

To run the script, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the rvkwasaut script:
   ```
   rvkwasaut -instance instance -user usrprf | -authlist none
   ```
   where *instance* is the instance to which you want are revoking authority, *usrprf* is the user profile from which you are revoking authority, and *authlist* is the authorization list from which you are revoking authority. You do not need to specify both the -user and -authlist parameters, but you must specify at least one of them.

Syntax

The syntax of the script is:

```
rvkwasaut -instance instance  -user usrprf | -authlist none
 [ -object path ] [ -recursive ] [ -verbose ] [ -help ]
```

Parameters

The parameters of the script are:

- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance to which you are revoking authority.
- **-user**
  The value *usrprf* specifies the i5/OS user profile from which you are revoking authority. To revoke authority from multiple user profiles, specify all of the user profiles with a single -user parameter. Enclose the list of profiles in double quotation marks (“). For example, to revoke authority from usrprf1 and usrprf2, specify `-user "usrprf1 usrprf2"`. You must specify -user, -authlist, or both.
- **-authlist**
  The value none specifies that the current authorization list associated with the object should be removed. If you use the -authlist parameter, you must specify the value *none*. You must specify either -user, -authlist, or both.
- **-object**
  This is an optional parameter. The value *path* specifies the subdirectory or partially qualified object name to which you are revoking authority. The instance root is prepended to the value to get the fully-qualified path. If you do not specify this parameter, the default value is the instance root. To revoke authority to multiple objects, you must run the script for each object.
- **-recursive**
  This optional parameter specifies whether to revoke authority to all subdirectories. If you do not specify this parameter, authority is revoked only to the object specified with the -object parameter, or the instance root directory if the -object parameter is not specified. This parameter applies to all objects specified with -object parameters.

- **-verbose**
  This optional parameter turns on verbose messages, which can be helpful if you need to debug the script.
- **-help**
  This optional parameter displays the help message. If you specify this parameter, the script ignores all other parameters.

Examples

In this example, the user profile jsmith no longer has authority to the instance devinst.

```
rvkwasaut -instance devinst -user jsmith -recursive
```

In this example, the user profile jsmith no longer has authority to the installedApps subdirectory and all nested objects in the installedApps subdirectory.

```
rvkwasaut -instance devinst -object installedApps -user jsmith -recursive
```

Encode password data

The EncAuthDataFile script encodes password data that is stored in a text file. The text file's name is defined by the system property WAS_AuthDataFile and has records in this format:

```
alias1,user1,pw1
```

For information about encoding algorithms, see Password encoding in the *Security* topic.

Authority

To run the EncAuthDataFile script, your user profile must have *ALLOBJ authority.

Syntax

The syntax of the EncAuthDataFile script is:

```
EncAuthDataFile input_file output_file [ -instance instance ]
```

Parameters

The parameters of the EncAuthDataFile script are:

- ***input_file***
  This is a required parameter. The value *input_file* specifies the fully qualified name of the authentication data file that the script reads.
- ***output_file***
  This is a required parameter. The value *output_file* specifies the fully qualified name of the authentication data file to which the script writes the encoded data.
- **-instance**
  This is an optional parameter. The value *instance* specifies an application server instance name. The script uses the password encoding algorithm that it retrieves from the specified instance. If -instance is not specified the default instance is used.

Encode passwords in properties files

Use the PropFilePasswordEncoder script to encode passwords in properties files.

Authority

To run this script, your user profile must have *ALLOBJ authority.

Syntax

```
PropFilePasswordEncoder fileName { passwordPropertiesList
  | -SAS } [ -instance instance ] [ -help | -? ]
```

Note: You must specify either the *passwordPropertiesList* parameter or the -SAS parameter.

Parameters

- ***fileName***
  This required parameter specifies the name of file in which passwords are encoded.
- ***passwordPropertiesList***
  This parameter is required if you are encoding passwords in the soap.client.props file. Specify a one or more password properties that you want to encode.
- **-SAS**
  This parameter is required if you are encoding passwords in the sas.client.props file.
- **-instance**
  This is an optional parameter. The value *instance* specifies an application server instance name. The script uses the password encoding algorithm that it retrieves from the specified instance. If -instance is not specified the default instance is used.
- **-help** or **-?**
  If you specify this parameter, the script ignores all other parameters and displays usage text.

Examples

This command encodes the passwords in the sas.client.props file for the default WebSphere Application Server - Express instance:

```
/QIBM/ProdData/WebASE51/ASE/bin/PropFilePasswordEncoder
  /QIBM/UserData/WebASE51/ASE/default/properties/sas.client.props -SAS
```

This command encodes passwords in the soap.client.props file for the default Websphere Application Server - Express instance:

```
/QIBM/ProdData/WebASE51/ASE/bin/PropFilePasswordEncoder
  /QIBM/UserData/WebASE51/ASE/default/properties/soap.client.props
  com.ibm.SOAP.loginPassword,com.ibm.ssl.keyStorePassword,com.ibm.ssl.trustStorePassword
```

Note: These commands are wrapped for display purposes. When you run the PropFilePasswordEncoder script, enter the command as a single line.

Advanced application server settings

The topics listed here describe advanced application server settings.

"Change application server ports with the chgwassvr script" on page 50
This topic describes how to use the chgwassvr script in Qshell to change the ports that your application server uses. Under most circumstances, you do not need to change these ports.

"Change application server ports with the console and wsadmin" on page 52
This topic describes how to use the administrative console and wsadmin to change the ports that your application server uses. Under most circumstances, you do not need to change these ports.

"Manage thread pool settings for the Web container with the administrative console" on page 53
This topic describes how to use the administrative console to configure a thread pool for your application server.

"Administer Java virtual machine settings" on page 53
This topic describes how to use the administrative console and wsadmin to configure Java virtual machine settings for your application server.

“Configure name space bindings” on page 55
This topic describes how to use the administrative console and wsadmin to configure name space bindings for your application server.

“Administer shared libraries” on page 56
This topic describes how to administer access to shared libraries for your applications.

“Administer resource adapters” on page 58
A resource adapter (RAR) represents an archive file. This file contains code that implements a library and connects to an Enterprise Information System (EIS) backend database. Resource adapters are obtained from the provider of the EIS software. This topic describes how to administer resource adapters.

“Configure application servers for other language environments” on page 60
This topic describes how to configure individual application servers to support different language environments.

Change application server ports with the chgwassvr script

You can use the chgwassvr script in Qshell to change one or more port numbers of an application server. Unless there is a port conflict, you do not need to change these settings. If your application server is not running correctly, and you suspect that there may be a port conflict, see Troubleshooting before you change any of the port settings. For information on running scripts in Qshell, see “Configure Qshell to run WebSphere Application Server - Express scripts” on page 92.

Authority

To run this script, your user profile must have *ALLOBJ authority.

Usage

To run the script, follow these steps:
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the chgwassvr script:
   ```
   chgwassvr -instance instance -server server [options]
   ```
   where *instance* is the name of the instance that contains the server you want to change, *server* is the name of the server you want to change, and *options* specifies the changes you want to make.

Syntax

The syntax of the script is:
```
chgwassvr -instance instance -server servername [ -portblock portblock ]
 [ -transport -oldport oldvalue -newport newvalue ] [ -inthttp inthttpport ]
 [ -admin adminport ] [ -adminssl adminsslport ] [ -soap soapport ]
 [ -nameservice nameserviceport ] [ -sas sasserverport ]
 [ -csiv2server csiv2serverauthport ] [ -csiv2client csiv2clientauthport ]
 [ -verbose ] [ -help ]
```

Parameters

The parameters of the script are:
- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance that contains the application server you want to change.

- **-server**
  This is a required parameter. The value *servername* specifies the name of the application server to change.
- **-portblock**
  This is an optional parameter. The value *portblock* specifies the first number of a block of port numbers that your instance uses. If you specify this parameter, the script changes all of the port numbers for your application server. If you do not specify this parameter, the port numbers for your application server are not changed, unless you specify a port parameter to change (see port parameters below). You can use the Work with TCP/IP Network Status (NETSTAT *CNN) command to display a list of port numbers that are currently being used.

  Note: A WebSphere Application Server - Express instance uses several ports for various functions. When you change an application server's properties, ports are assigned based on these ordered conditions:

  1. **Specific port parameters**
     If you specify values for specific port parameters, the script uses those values. Specific port parameters are -inthttp, -admin, -soap, and -nameservice.
  2. **The -portblock parameter**
     Services for which you have not specified a port number are assigned ports sequentually starting with the value of the -portblock parameter. If a script encounters a port that is in use, it skips that port number and continues with the next unused port.
  3. **Current values**
     If -portblock is not specified, any services for which you have not specified a port parameter retain their current value.
- **-transport**
  This is an optional parameter set. The value *oldvalue* specifies the port number of the Web container transport that you want to change. The value *newvalue* specifies the new port number that you want to assign to the Web container transport. If the Web Container transport is also specified in the host alias for a virtual host, the port for the host alias is also updated.

  Note: Use the -transport parameter set instead of the -admin, -adminssl, or -inthttp parameters. These parameters are deprecated.
- **-inthttp**
  This is an optional parameter. The value *inthttpport* specifies the port number on which the Web container listens for requests from the Web server. If neither the -portblock parameter nor the -inthttp parameter is specified, this port is not changed. See the Note (page 51) on the -portblock parameter for more information.

  Note: This parameter is deprecated. Use the -transport -oldport *oldvalue* -newport *newvalue* parameter set to change this value. For exapmle, to change the internal HTTP server port for the default WebSphere Application Server - Express instance from 9080 to 9081, run this command at the Qshell prompt:

  ```
  chgwassvr -server server1 -transport -oldport 9080 -newport 9081
  ```
- **-admin**
  This is an optional parameter. The value *adminport* specifies the port number to use for the WebSphere administrative console. If neither the -portblock parameter nor the -admin parameter is specified, this port is not changed. See the Note (page 51) on the -portblock parameter for more information.

  Note: This parameter is deprecated. Use the -transport -oldport *oldvalue* -newport *newvalue* parameter set to change this value. For exapmle, to change the administrative console port for the default WebSphere Application Server - Express instance from 9090 to 9091, run this command at the Qshell prompt:

  ```
  chgwassvr -server server1 -transport -oldport 9090 -newport 9091
  ```
- **-adminssl**
  This is an optional parameter. The value *adminportssl* specifies the port number to use for the secure

communications with WebSphere administrative console. If neither the -portblock parameter nor the -adminssl parameter is specified, this port is not changed. See the Note (page 51) on the -portblock parameter for more information.

Note: This parameter is deprecated. Use the -transport -oldport *oldvalue* -newport *newvalue* parameter set to change this value. For exapmle, to change the administrative console SSL-enabled port for the default WebSphere Application Server - Express instance from 9043 to 9044, run this command at the Qshell prompt:

```
chgwassvr -server server1 -transport -oldport 9043 -newport 9044
```

- **-soap**
  This is an optional parameter. The value *soapport* specifies the port number to use for Simple Object Access Protocol (SOAP). If neither the -portblock parameter nor the -soap parameter is specified, this port is not changed. See the Note (page 51) on the -portblock parameter for more information.
- **-nameservice**
  This is an optional parameter. The value *nameserviceport* specifies the port number to use for name service (or RMI connector) port. If neither the -portblock parameter nor the -nameservice parameter is specified, this port is not changed. See the Note (page 51) on the -portblock parameter for more information.
- **-sas**
  This is an optional parameter. The value *sasserverport* specifies the port on which the Secure Association Services (SAS) listen for inbound authentication requests. If the -sas parameter is not specified, this port is not changed. This port is specified by the SAS_SSL_SERVERAUTH_LISTENER_ADDRESS property in serverindex.xml.
- **-csiv2server**
  This is an optional parameter. The value *csiv2serverauthport* specifies the port on which the Common Secure Interoperability Version 2 (CSIV2) Service listens for inbound server authentication requests. If the -csiv2server parameter is not specified, this port is not changed. This port is specified by the CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS property in serverindex.xml.
- **-csiv2client**
  This is an optional parameter. The value *csiv2clientauthport* specifies the port on which the Common Secure Interoperability Version 2 (CSIV2) Service listens for inbound client authentication requests. If the -csiv2client parameter is not specified, this port is not changed. This port is specified by the CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS property in serverindex.xml.
- **-verbose**
  This optional parameter turns on verbose messages, which can be helpful if you need to debug the script.
- **-help**
  This optional parameter displays the help message. If you specify this parameter, the script ignores all other parameters.

Examples

```
chgwassvr -instance devinst -server devinst -portblock 11400
```

In this example, the ports assigned to the application server devinst in the instance devinst are changed.

```
chgwassvr -instance devinst -server devinst -transport -oldvalue 9090 -newvalue 12240
```

In this example, the WebSphere administrative console port for the application server devinst in the instance devinst is changed from 9090 to 12240.

Change application server ports with the console and wsadmin

This topic describes how to change port numbers in the serverindex.xml file. Unless there is a port conflict, you do not need to change these settings. If your application server is not running correctly, and you suspect that there may be a port conflict, see Troubleshooting before you change any of the port settings.

- Use the administrative console to change application server ports (page 53)
- Use wsadmin to change application server ports (page 53)

Use the administrative console to change application server ports

1. "Start the WebSphere administrative console" on page 83.
2. Expand **Servers** and click **Application Servers**.
3. Click the name of your application server.
4. On the application server's page, click **End Points**.
5. Click the name of the port that you want to change.
6. Make your changes.
7. Click **Apply** or **OK**.
8. "Save the application server configuration" on page 84.

Use wsadmin to change application server ports

Use the commands in these examples to modify the serverindex.xml file:

- **BOOTSTRAP_ADDRESS**
  An attribute of the NameServer object that exists inside the server. It is used by the naming client to specify the naming server to look up the initial context. To modify its end point, obtain the ID of the NameServer object and run a modify command. For example:
  ```
  set server [$AdminConfig getid /Server:myAppSvr/]
  set ns [$AdminConfig list NameServer $server]
  $AdminConfig modify $ns {{BOOTSTRAP_ADDRESS {{host myHost} {port myPort}}}}
  ```
  where *myAppSvr* is the name of your application server, *myHost* is the name of your iSeries server, and *myPort* is the name of the port that you want to assign to the name server.
- **SOAP_CONNECTOR-ADDRESS**
  An attribute of the SOAPConnector object that exists inside the server. It is the port that is used by the HTTP transport for incoming SOAP requests. To modify its end point, obtain the ID of the SOAPConnector object and run a modify command. For example:
  ```
  set server [$AdminConfig getid /Server:myAppSvr/]
  set soap [$AdminConfig list SOAPConnector $server]
  $AdminConfig modify $soap {{SOAP_CONNECTOR_ADDRESS {{host myHost} {port myPort}}}}
  ```
  where *myAppSvr* is the name of your application server, *myHost* is the name of your iSeries server, and *myPort* is the name of the port that you want to assign to the SOAP connector.

Manage thread pool settings for the Web container with the administrative console

Thread pools allow components of the application server to reuse threads. eliminate the need to create new threads at runtime. Using a thread pool reduces the amount of resources required to create new threads at run time.

1. "Start the WebSphere administrative console" on page 83.
2. Expand **Servers** and click **Application Servers**.
3. Click the name of your application server.
4. On the application server's page, click **Web Container**.
5. On the Web Container page, click **Thread Pool**.
6. Configure the thread pool settings.
7. Click **Apply** or **OK**.
8. "Save the application server configuration" on page 84.

Administer Java virtual machine settings

The Java virtual machine settings for your application server specify Java virtual machine system properties that your applications use.

You can use the administrative console and wsadmin to configure Java virtual machine settings.
- Use the administrative console to administer Java virtual machine settings (page 54)
- Use wsadmin to administer Java virtual machine settings (page 54)

Use the administrative console to administer Java virtual machine settings

To administer Java virtual machine settings with the administrative console, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. Expand **Servers** and click **Application Servers**.
3. Click the name of your application server.
4. On the application server's page, click **Process Definition**.
5. On the Process Definition page, click **Java Virtual Machine**.
6. Configure settings for the Java virtual machine.
7. Click **Apply** or **OK**.
8. "Save the application server configuration" on page 84.

Use wsadmin to administer Java virtual machine settings

To administer Java virtual machine settings with wsadmin, follow these steps:
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the application server and assign it to the server variable:
   ```
   set server [$AdminConfig getid
   /Cell:myCell/Node:myNode/Server:myAppSvr/]
   ```
   where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, and *myAppSvr* is the name of your application server.

   Note: This command has been wrapped for display purposes.
5. Run this command to identify the Java virtual machine for your application server and assign it to the jvm variable:
   ```
   set jvm [$AdminConfig list JavaVirtualMachine $server]
   ```
6. Use the attributes command to display a list of the attributes that you can set for the Java virtual machine:
   ```
   $AdminConfig attributes JavaVirtualMachine
   ```
   The command displays this output:
   ```
   "bootClasspath String*"
   "classpath String*"
   "debugArgs String"
   "debugMode Boolean"
   "disableJIT Boolean"
   "executableJarFileName String"
   "genericJvmArguments String"
   "hprofArguments String"
   "initialHeapSize Integer"
   "maximumHeapSize Integer"
   "osName String"
   "runHProf Boolean"
   "systemProperties Property(TypedProperty)*"
   "verboseModeClass Boolean"
   "verboseModeGarbageCollection Boolean"
   "verboseModeJNI Boolean"
   ```

7. Modify the Java virtual machine attributes. For example, to turn on debugging, run this command:
   ```
   $AdminConfig modify $jvm {{debugMode true} {debugArgs "-Djava.compiler=NONE
   -Xdebug -Xnoagent -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=7777"}}
   ```
8. Run this command to save your changes:
   ```
   $AdminConfig save
   ```

Configure name space bindings

The application server name space consists of references to objects that the application server uses. A name space binding is a reference to a particular object. Because the name of an object in the name space does not change, you can change the names of the resources that your applications use, and the applications can still connect to those resources. See Java Naming and Directory Interface (JNDI) for more information on naming.

You can use the administrative console and wsadmin to configure name space bindings.

- Use the administrative console to configure name space bindings (page 55)
- Use wsadmin to configure name space bindings (page 55)

Use the administrative console to configure name space bindings

To configure name space bindings with the administrative console, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. Expand **Environment —> Naming** and click **Name Space Bindings**.
3. Verify that the scope is set to the server level.
4. Add, modify, or remove a name space binding.
   - To add a name space binding, follow these steps:
     a. Click **New**.
     b. Specify the type of binding that you want to create and click **Next**.
     c. Configure the settings for the binding and click **Next**.
     d. Review the settings and click **Finish**.
   - To modify a name space binding, follow these steps:
     a. Click the name of the the binding that you want to modify.
     b. Make your changes.
     c. Click **Apply** or **OK**.
   - To remove a name space binding, follow these steps:
     a. Select the binding that you want to remove.
     b. Click **Delete**.
5. "Save the application server configuration" on page 84

Use wsadmin to configure name space bindings

To configure name space bindings with wsadmin, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the application server and assign it to the server variable:
   ```
   set server [$AdminConfig getid /Cell:myCell/Node:myNode/Server:myAppSvr/]
   ```

where *myCell* is the name of the cell that contains your application server, *myNode* is the name of the node that contains your application server, and *myAppSvr* is the name of your application server.

5. Add a new name space binding on the server.
   - To configure a string type name space binding, run this command:
     ```
     $AdminConfig create StringNameSpaceBinding $server
      {{name myBinding1} {nameInNameSpace myBindings/myString}
      {stringToBind "myStringValue"}}
     ```
     where *myBinding1* is the name of the name space binding, *myBindings/myString* is the name of the binding as it appears in the name space, and *myStringValue* is the string that you want to bind.

     Note: This command has been wrapped for display purposes.
   - To configure a CORBA type name space binding, run this command:
     ```
     $AdminConfig create CORBAObjectNameSpaceBinding $server
      {{name myBinding2} {nameInNameSpace myBindings/myCORBA}
      {corbanameUrl corbaname:iiop:host.mycompany.com:port#stuff/myCORBAObject}}
     ```
     where *myBinding2* is the name of the name space binding, *myBindings/myCORBA* is the name of the binding as it appears in the name space, and the value of the corbanameUrl attribute is the URL for the object that you want to bind. Specifically, *host.mycompany.com* is the name of your iSeries server, *port* is the name service port for your application server, and *#stuff/myCORBAObject* is the CORBA object that you want to bind.

     Note: This command has been wrapped for display purposes.
   - To configure an indirect type name space binding, run this command:
     ```
     $AdminConfig create IndirectLookupNameSpaceBinding $server
      {{name myBinding3} {nameInNameSpace myBindings/myIndirect}
      {providerURL corbaloc::host.myCompany.com:port/NameServiceServerRoot}
      {jndiName jndiName}}
     ```
     where *myBinding3* is the name of the name space binding, *myBindings/myIndirect* is the name of the binding as it appears in the name space, the value of the providerURL attribute is the URL to the CORBA name service, and *jndiName* is the JNDI name of the object that you want to bind.

     Note: This command has been wrapped for display purposes.
6. Run this command to save your changes:
   ```
   $AdminConfig save
   ```

Administer shared libraries

WebSphere Application Server - Express uses shared libraries to define code that is used by your enterprise applications, but not packaged within those applications. For example, a utility library, such as IBM Toolbox for Java classes, could be packaged as a shared library. Shared libraries can be associated with an enterprise application or an application server. If you assoicate a shared library with an application server, it is accessible to all of the enterprise applications deployed on that server.

Shared library files in WebSphere Application Server consist of a symbolic name, a classpath, and a path for loading JNI libraries.

You can define a shared library at the cell, node, or server level. Defining a library at one of the three levels does not cause the library to be placed into the application server's classloader. To make the classes in the library available to an application or server, you must create an association between the library and the application or server.

A separate classloader is used for shared libraries that are associated with an application server. This classloader is the parent of the application classloader, and the WebSphere Application Server extensions classloader is its parent. Shared libraries that are associated with an application are loaded by the application classloader. For more information about classloaders, see Classloader hierarchy.

For information on administering shared libraries, see these sections:
- Create shared libraries (page 57)

- Modify shared libraries (page 57)
- Remove shared libraries (page 57)
- Associate a shared library with an application server (page 57)
- Associate a shared library with an application (page 58)

Create shared libraries

To create a shared library, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Shared Libraries**.
3. On the **Shared Libraries** page, specify the scope for which you want to define a library and click **Apply**.
4. Click **New**.
5. Specify a name and classpath for the shared library. You can also specify other properties on this page.
6. Click **OK**.
7. "Save the application server configuration" on page 84.

Modify shared libraries

To modify a shared library, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Shared Libraries**.
3. On the **Shared Libraries** page, specify the scope contains the library that you want to modify and click **Apply**.
4. Click the name of the shared library that you want to modify.
5. Make your changes.
6. Click **OK**.
7. "Save the application server configuration" on page 84.

Remove shared libraries

To remove a shared library, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Environment** and click **Shared Libraries**.
3. On the **Shared Libraries** page, specify the scope for that contains the library that you want to remove and click **Apply**.
4. Select the checkbox for the shared library that you want to remove.
5. Click **Delete**.
6. "Save the application server configuration" on page 84.

Associate a shared library with an application server

To associate a shared library with an application server, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Servers** and click **Application Servers**.
3. On the **Application Servers** page, click the name of the server to which you want to add a shared library.
4. On the application server's detail page, click **Classloader**.

5. On the **Classloader** page, click **New**.
6. Select the **Classloader** mode and click **OK**.
7. On the **Classloader** page, click the classloader that you created. If this is the first classloader that you create for your application server, it is named classloader_1.
8. On the classloader's detail page, click **Libraries**.
9. On the **Libraries** page, click **Add**.
10. Select the shared library that you want to associate with your application server.
11. Click **OK**.
12. "Save the application server configuration" on page 84.

To remove an association to a shared library, follow these steps:
1. On the **Libraries** page, select the checkbox for the library that you want to remove.
2. Click **Remove**.
3. "Save the application server configuration" on page 84.

Associate a shared library with an application

To associate a shared library with an application, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Applications** and click **Enterprise Applications**.
3. On the **Enterprise Applications** page, click the name of the application to which you want to add a shared library.
4. On the application's detail page, click **Libraries**.
5. On the **Libraries** page, click **Add**.
6. Select the shared library that you want to associate with your application.
7. Click **OK**.
8. "Save the application server configuration" on page 84.

To remove an association to a shared library, follow these steps:
1. On the **Libraries** page, select the checkbox for the library that you want to remove.
2. Click **Remove**.
3. "Save the application server configuration" on page 84.

Administer resource adapters

A resource adapter (RAR) represents an archive file. This file contains code that implements a library and connects to an Enterprise Information System (EIS) backend database. Typically, a single resource adapter can only connect to one type of enterprise information system (EIS) but it can support many different configurations for connections to that EIS. The resource adapter has many configuration properties that are defined in the J2C specification and set by the vendor who supplies the code.

For help with resource adapter settings, see these topics:
- Resource adapter settings

- J2C connection factory settings

For information on configuring, modifying, and removing resource adapters, see these sections:

- Install Java 2 Connector RAR files (page 59)
- Create resource adapters (page 59)
- Modify resource adapters (page 59)
- Remove resource adapters (page 59)

Install Java 2 Connector RAR files

You can use the administrative console to install a J2EE Connector Architecture (JCA) connector and create a resource adapter for it.

To install a Java 2 Connector RAR file, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Resources** and click **Resource Adapters**.
3. On the **Resource Adapters** page, click **Install RAR**.

   Note: When you install a RAR file with the administative console, the scope that you specify on the **Resource Adapters** page does not affect where the RAR file is installed. You can install RAR files only at the node level, and the node the file is installed on is determined by the scope on the **Install RAR** page. The scope you set on the **Resource Adapters** page determines the scope of the new resource adapters, which you can create at the server, node, or cell level.
4. Specify the RAR file that you want to install, or click **Browse...** to select the file.
5. Click **Next**.
6. Specify the resource adapter name. You can also specify additional properties on this page.
7. Click **OK**.

Create resource adapters

If a JCA connector is already installed on your system, and you want to create a new resource adapter for the connector, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Resources** and click **Resource Adapters**.
3. On the **Resource Adapters** page, click **New**.
4. Specify a name and archive path. You can also specify additional properties on this page.
5. Click **Apply**.
6. After you create the resource adapter, you can configure J2C connection factories for it.
7. "Save the application server configuration" on page 84.

Modify resource adapters

To modify a resource adapter, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **Resources** and click **Resource Adapters**.
3. On the **Resource Adapters** page, select the name of the resource adapter that you want to modify.
4. Make your changes.
5. Click **OK**.
6. "Save the application server configuration" on page 84.

Remove resource adapters

To remove a resource adapter, follow these steps:

1. "Start the WebSphere administrative console" on page 83.

2. In the topology tree, expand **Resources** and click **Resource Adapters**.
3. On the **Resource Adapters** page, select the checkbox for the name of the resource adapter that you want to remove.
4. Click **Delete**.
5. "Save the application server configuration" on page 84.

Configure application servers for other language environments

You can configure individual application servers to run with different globalization settings to support different language environments. The topics below discuss how you can configure the application server to run with the language environment attributes which you require, and how you can configure the application server to use a specific National Language Version (NLV) when you have multiple WebSphere Application Server - Express NLVs installed.

- **Configure the language environment attributes**
  You can configure the application server to run with the appropriate language environment attributes, such as the coded character set identifier (CCSID) and country or region identifier, for the environment you wish to create. The QEJBSVR user profile settings are the basis for the application server job attributes, and the job attributes determine the properties for the Java$^{(TM)}$ virtual machine environment.

  Note: Japanese CCSID 5026 is not supported by WebSphere Application Server - Express. 5035 is the recommended CCSID for this environment. The QEJB and QEJBSVR user profiles default to the system CCSID setting. If the QCCSID system value is set to 5026, the CCSID attribute of the QEJB user profile must be changed to prevent the WebSphere Application Server - Express Monitor and Administration server jobs from trying to start with a CCSID of 5026. The QEJBSVR user profile, and any other user profiles used to run WebSphere Application Server - Express instances, must also be changed.

  By default, each application server runs under the QEJBSVR user profile. To change the language environment settings you can either modify the QEJBSVR profile and set the attributes specifically or you can create a new profile which has QEJBSVR as its group profile and which has the appropriate language environment settings. If you choose to create a new profile, you must also register it for use with WebSphere Application Server - Express. For more information, see Run application servers under specific user profiles.
- **WebSphere Application Server - Express product settings**
  A second consideration is the national language version (NLV) of the WebSphere Application Server - Express product. The NLV setting only causes the library list to be changed to include the appropriate QSYS*xxxx* library, where *xxxx* is the language feature for which you wish to display messages. The NLV setting does not affect the WebSphere administrative console or any non-iSeries message logged by the WebSphere Application Server - Express runtime."

  Multiple NLVs can be installed at the same time. If the language version that you want to use is the same as the primary language of the system, no additional configuration is needed. If the language version you want to use is a secondary language, configure the application server by setting the os400.websphere.nlv property:
  1. "Start the WebSphere administrative console" on page 83
  2. In the topology tree, expand **Servers** and click **Application Servers**.
  3. On the **Application Servers** page, click the name of the server for which you wish to change language version.
  4. On the **Configuration** tab, and click **Process Definition —> Java Virtual Machine —> Custom Properties**.
  5. On the **Custom Properties** page, click **New**.
  6. Specify these values for the custom property settings:
     - **Name:** os400.websphere.nlv
     - **Value:** *xxxx*

     where *xxxx* is the language feature you want to use to display messages. For example, to specify the Japanese language, set the Value as 2962. You can also specify a description for this property.

7. Click **OK**.
8. "Save the application server configuration" on page 84.

When an application server has the os400.websphere.nlv property set, the corresponding QSYS29*xx* library is added to the beginning of the library list and iSeries messages sent by the application server to the joblog are used accordingly.

Deploy and start a new application

Before you can deploy and start a new application, you must create and configure an application server instance. If you have not already done so, see "Create a new application server" on page 2 for information.

An application can consist of one or more Web modules packaged in an Enterprise Archive (EAR) file, or a single Web module packaged as a stand-alone Web Archive (WAR) file. A Web module is a logical piece of an application, containing objects like Java servlets, JavaServer Pages (JSP) files, and HTML pages. The file must exist in the location specified before the wizard can proceed.

To deploy a new application, follow these steps:

1. Start the HTTP Server Administration interface.
2. Click the **Manage** tab.
3. Expand **Tasks and Wizards**.
4. Click **Install New Application**.
5. Select one of these options:
   - **Application is contained in an EAR file**
     a. Click **Browse** or specify the fully qualified path name of the EAR file.

        Note: It is not necessary to specify the context root for an EAR file. The context root is specified in the EAR file.
   - **Application is contained in a WAR file**
     a. Click **Browse** or specify the fully qualified path name of the WAR file.
     b. Specify the root of the context in the **Context root** field. The context root is the path used in a client request to access the application resources.
6. Click **Next**.
7. Specify a name for the application in the **Application name** field. The default value is the name of the file.

   Note: The application name value is specified within the file application. If not specified, the application name value defaults to the name of the application file.
8. **Optional:** Select **Pre-compile JSPs** to compile JSP files into Java servlet code during the installation of the application. If the JSP file is pre-compiled, the application runs faster the first time you start it. However, depending on the number of JSP files in the application and the size of your system, pre-compiling all JSP files can cause the install of the application to take a significant amount of time.
9. Click **Next**.
10. Select a virtual host to map to a Web module from the **Map web modules to virtual hosts** table. The virtual host determines the IP Address or hostname that is used to access your application. See "Manage virtual hosts for your application server" on page 26 for information on creating new virtual hosts.
11. Click **Next**.
12. The summary page lists all of the choices you have made in the wizard. If any of the information displayed is incorrect, click **Back** until you reach the wizard form with the incorrect information and make your corrections. Click **Finish** to complete the wizard.

13. After you deploy the application,

    "Manage installed applications for your application server"
    This topic describes how to start and stop applications, and how to enable and disable applications.

    "Install and uninstall applications with the WebSphere administrative console" on page 64
    This topic describes how to use the administrative console to install and uninstall applications in your application server.

    "Install and uninstall applications with wsadmin" on page 66
    This topic describes how to use wsadmin to install and uninstall applications in your application server.

    "Start and stop applications with the WebSphere administrative console" on page 67
    This topic describes how to use the administrative console to start and stop applications.

    "Start and stop applications with wsadmin" on page 67
    This topic describes how to use wsadmin to start and stop applications.

    "Use the EARExpander script to work with applications" on page 72
    This topic describes how to use the EARExpander script in Qshell to expand and collapse applications in EAR files.

    "Advanced application configuration" on page 68
    This topic describes advanced application settings that you can configure.

Manage installed applications for your application server

You can use the HTTP Server Adminsitration interface to manage the applications that are deployed in your application server instance. You can use the interface to perform the basic tasks required to manage applications.

To manage your installed applications on an application server, follow these steps:

1. Start the HTTP Server Administration interface.
2. Click the **Manage** tab.
3. Select an application server from the **Server** list.
4. Expand **Applications**.
5. Click **Manage Installed Applications**.

From the Manage Installed Applications form, you can do the following:

- Start or stop an application (page 62)
- Enable or disable an application (page 63)
- Edit the virtual host mapped to a Web module (page 63)
- Update an application (page 63)
- Uninstall an application (page 63)

Start or stop an application

1. Select an application from the **Installed applications** table.
2. Click **Start** to start an application.
3. Click **Stop** to stop an application.

Note: The start button is not available if the application is running or if it is disabled.

Enable or disable an application

1. Select an application from the **Installed applications** table.
2. Click **Properties**.
3. Click the **General** tab in the Application Properties form.
4. Select one of these options from the **Application enablement** field.
   - **Enabled**
     Select **Enabled**. Enabled applications start automatically when you start the application server.
   - **Disabled**
     Select **Disabled**. Disabled applications do not start when you start the application server. You cannot run an application that is disabled.
5. Click **Apply** or **OK** to save your changes.

Edit the virtual host mapped to a Web module

1. Select an application from the **Installed applications** table.
2. Click **Properties**.
3. Click the **Virtual Host Mapped to Web Modules** tab in the Application Properties form.
4. Select a Web module from the **Web modules and corresponding virtual hosts for application** table.
5. Select a virtual host to map to a Web module. The virtual host determines the IP address or hostname that is used to access your application. See "Manage virtual hosts for your application server" on page 26 for information on creating new virtual hosts.
6. Click **Apply** or **OK** to save your changes.

Update an application

Updates to applications redeploys it for your application server. It does not change configuration attributes.

1. Select an application from the **Installed applications** table.
2. Click **Stop** if the application is running.
3. Click **Update**.
4. Select select one of these options.
   - **Application is contained in an EAR file**
     a. Click **Browse** or specify the fully qualified path name of the EAR file on your iSeries.
   - **Application is contained in a WAR file**
     a. Click **Browse** or specify the fully qualified path name of the WAR file on your iSeries.
   - **Optional**: Select **Pre-compile JSPs** to compile JSP files into Java servlet code during the installation of the application. If the JSP files do not need to be compiled at runtime, the application server performance can improve. However, depending on the number of JSP files in the application and the size of your system, pre-compiling all JSP files can cause the install of the application to take a significant amount of time.
5. Click **Update**.

Uninstall an application

1. Select an application from the **Installed applications** table.
2. Click **Stop** if the application is running.
3. Click **Uninstall.**
4. Click **OK**.
5. Click **Refresh** to update the **Installed applications** table.

Install and uninstall applications with the WebSphere administrative console

You can use the WebSphere administrative console to install and uninstall enterprise applications on your application server.

- Install applications with the administrative console (page 64)
- Uninstall applications with the administrative console (page 66)

Install applications with the administrative console

To install an application, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. Expand **Applications** and click **Install New Application**.
3. Specify the path of the application that you want to install.
   - If the application is stored on your workstation, select **Local path**.
   - If the application is stored on a node in your WebSphere Application Server - Express cell, select **Server path**. For a server path, specify the fully qualified path of the application. For example, if the EAR file myApp.ear for your application is located in the installableApps directory of the myAppSvr instance, specify:

     ```
     /QIBM/UserData/WebASE/ASE5/myAppSvr/installableApps/myApp.ear
     ```

     Note: When using the Server path option, the QEJBSVR user profile must have *RX authority to the EAR or WAR file and the directory where the file is located.
4. If the application is a WAR module, specify the context root.
5. Click **Next**.
6. Specify options for bindings and mappings.
7. Click **Next**.

   Note: WebSphere Application Server - Express reads the deployment descriptor of your application before it displays the next page. It uses the information from the deployment descriptor to determine which information you need to provide before you install the application. As a result, you may not see all of the panels described here.
8. **Provide options to perform the installation**
   On this panel, you can specify several general options that determine how WebSphere Application Server - Express installs your application.
   a. If you want WebSphere Application Server - Express to precompile JavaServer Pages (JSP) files as part of the installation, select **Pre-compile JSP**. If you do not select this option, JSPs are compiled the first time they are accessed.
   b. Specify the directory where you want to install the application EAR file. By default, the application is installed in the /QIBM/UserData/WebASE/ASE5/*instance*/installedApps/*node* directory, where *instance* is the name of your instance and *node* is the name of the node for your instance. It is recommended that you use the default value.
   c. Specify where the application configuration used during runtime comes from. If this option is not selected, configurations specified during application installation, and stored in the configuration files, are used. If this option is selected, the configuration specified during installation is ignored, and the settings in the original application are used. The default is not to use the binary configuration. It is recommended that you use the default setting.
   d. Specify a name for the application. Application names must be unique within a cell.

      Note: Because WebSphere Application Server - Express does not support enterprise beans, the **Deploy EJBs** option does not apply. Ignore this option.
   e. By default, WebSphere Application Server - Express creates MBeans for various resources (such as servlets or JSP files) within an application when the application is started. These MBeans allow you to use the the HTTP Server Adminisration interface and the wsadmin administrative tool to

manage your application. It is recommended that you use the HTTP Server Administration interface to manage applications. If you do not want to use this capability, deselect this option.

f. To enable class reloading when application files are updated, select **Enable class reloading**. If you enable class reloading, you can update your application and apply the changes without restarting the application server. In addition, WebSphere Application Server - Express periodically scans the application for updates. It may be convenient to have reloading turned on in a development environment. In a production environment, it is suggested that you disable class reloading, or that you specify a long reload interval.

g. If you enable class reloading, you can also specify the reload interval (in seconds). This value specifies how often the application server scans the application's file system for updated files. The default value is specified by the reload interval attribute in the IBM extension (META-INF/ibm-application-ext.xmi) file of the EAR file or the IBM extension file for each Web module in the EAR file.

9. **Map resource references to resources**
   If your application defines resource references, specify JNDI names for the resources that represent the logical names defined in resource references. You must bind all of the resource references to a resource before you click **Finish**. The resources are defined in your WebSphere Application Server - Express configuration.

10. **Map virtual hosts for web modules**
    If your application contains Web modules, you must map each Web module to a virtual host. Select a virtual host to map to a Web module defined in the application. The port number specified in the virtual host definition is used in the URL that is used to access objects such as servlets and JSP files in the Web module.

    Note: You should never select the admin_host virtual host. This is a special virtual host used by the WebSphere administrative console application.

11. **Map security roles to users/groups**
    If the application has security roles defined in its deployment descriptor, you can specify users and groups to map to each role. Select the checkbox beside Role to select all of the roles or select individual roles. You can map predefined users such as **Everyone** or **All Authenticated** to each role. To select specific users or groups from the user registry:

    a. Select a role and click **Lookup users** or **Lookup groups**.

    b. On the Lookup users/groups panel, enter search criteria to view a list of users or groups from the user registry.

    c. Select one or more users or groups from the results displayed.

    d. Click **OK** to map the selected users or groups to the selected role or roles.

12. **Map RunAs roles to user**
    If the application's deployment descriptor defines one or more RunAs roles, specify the RunAs user name and password for every RunAs role. RunAs roles are used by enterprise beans that must run as a particular role while interacting with another enterprise bean. Select the checkbox for the role or roles that you want to configure. After you select a role, enter values for the user name and password, then click **Apply**.

13. **Mapping Resource Environment References to Resources**
    If your application contains resource environment references, specify JNDI names of resources that map to the logical names defined by the references. If each resource environment reference does not have a resource associated with it, the installation wizard displays a validation error when you click **Finish**.

14. **Replacing RunAs System to RunAs Roles**
    If your application defines RunAs Identity as System Identity, you can optionally change it to RunAs role and specify a user name and password for the RunAs role specified. Selecting System Identity implies that the invocation is done using the WebSphere Application Server - Express security server ID and should be used with caution.

15. On the **Summary** panel, verify that the installation settings are correct. For the Cell/Node/Server item, click **Click here** to verify the settings.
16. Click **Finish**.
17. "Save the application server configuration" on page 84. The application is registered with the administrative configuration and application files are copied to the target directory when you save the configuration.

Uninstall applications with the administrative console

To uninstall an application with the administrative console, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. Expand **Applications** and click **Enterprise applications**.
3. Select the check box for the application or applications that you want to remove.
4. Click **Uninstall**.
5. "Save the application server configuration" on page 84.

Install and uninstall applications with wsadmin

You can use wsadmin to deploy applications into your application server. The wsadmin tool supports non-interactive installation, in which you run a single command that specifies the application to install, and interactive installation, in which you are prompted for information about the installation. You can also use wsadmin to uninstall applications from your application server.
- Install applications with wsadmin (page 66)
- Uninstall applications with wsadmin (page 66)

Install applications with wsadmin

To use the wsadmin tool to deploy applications into your application server, follow these steps:
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run one of these commands:
   - This command uses the EAR file and the command option information to install the application:
     ```
     $AdminApp install earfile {-server myAppSvr}
     ```
     where *earfile* is the fully qualified path of the EAR file that you want to install and *myAppSvr* is the name of your application server instance.
   - This command starts an interactive installation that prompts you through a series of installation tasks:
     ```
     $AdminApp installInteractive earfile
     ```
     where *earfile* is the fully qualified path of the EAR file that you want to install.
5. Run this command to save your changes:
   ```
   $AdminConfig save
   ```

Uninstall applications with wsadmin
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to uninstall an application:

```
$AdminApp uninstall myApp
```

where *myApp* is the name of the application that you want to uninstall.

Note: Specify the name of the application, not the name of the EAR or WAR file.

5. Run this command to save your changes:

```
$AdminConfig save
```

Start and stop applications with the WebSphere administrative console

You can use the administrative console to start and stop your applications.

- Start an application (page 67)
- Stop an application (page 67)

Start an application

To start an application, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. Expand **Applications** and click **Enterprise applications**.
3. Select the check box for the application or applications that you want to start.
4. Click **Start**.

Stop an application

To stop an application, follow these steps:

1. "Start the WebSphere administrative console" on page 83.
2. Expand **Applications** and click **Enterprise applications**.
3. Select the check box for the application or applications that you want to stop.
4. Click **Stop**.

Start and stop applications with wsadmin

You can use wsadmin to start and stop applications in your application server.

- Start applications with wsadmin (page 67)
- Stop applications with wsadmin (page 67)

Start applications with wsadmin

To start an application with wsadmin, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:

   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```

3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the application manager MBean for the server where the application resides and assign it the appManager variable:

   ```
   set appManager [$AdminControl queryNames type=ApplicationManager,*]
   ```

   This command returns the application manager MBean.
5. Run this command to start the application:

   ```
   $AdminControl invoke $appManager startApplication myApp
   ```

   where *myApp* is the name of the application that you want to start.

Stop applications with wsadmin

To stop applications with wsadmin, follow these steps:
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the application manager MBean for the server where the application resides, and assign it to the appManager variable:
   ```
   set appManager [$AdminControl queryNames type=ApplicationManager,*]
   ```
   This command returns the application manager MBean.
5. You can stop a single application or stop all of the applications that are running in your application server.
   - To stop a single application, run this command:
     ```
     $AdminControl invoke $appManager stopApplication myApp
     ```
     where *myApp* is the name of the application that you want to stop.
   - To stop all of the running applications in your application server, follow these steps:
     a. Run this command to query the running applications in the application server and assign the result to the apps variable:
        ```
        set apps [$AdminControl queryNames type=Application,*]
        ```
        This command returns a list of application MBeans.
     b. Run this command to stop all of the running applications:
        ```
        foreach app $apps {set appName [$AdminControl getAttribute $app name];
         $AdminControl invoke $appManager stopApplication $appName}
        ```
        Note: This command has been wrapped for display purposes.

Advanced application configuration

You can configure additional settings for applications. See these topics for more information:

"Administer session tracking for your application"
This topic describes how to configure session tracking for an application.

"Map virtual hosts for Web modules" on page 70
This topic describes how to map virtual hosts to individual Web modules on your application server.

"Configure session management for Web modules" on page 70
This topic describes how to configure session tracking for Web modules on your application server.

Administer session tracking for your application

Applications use sessions to associate HTTP requests from a single client. Session management settings determine how your application manages HTTP sessions.

You can configure session management for an application if you want that application to use different session management settings from the other settings for your application server. Settings for the application take precedence over settings for the application server.

You can use the administrative console and wsadmin to configure session tracking for your application.
- Use the administrative console administer session tracking for your application (page 68)
- Use wsadmin to administer session tracking for your application (page 69)

Use the administrative console administer session tracking for your application

1. "Start the WebSphere administrative console" on page 83.

2. Expand **Applications** and click **Enterprise Applications**.
3. Click the name of the application for which you want to manage session tracking.
4. On the application's page, click **Session Management**.
5. Configure session management settings.
6. Click **Apply** or **OK**.
7. "Save the application server configuration" on page 84.

Use wsadmin to administer session tracking for your application

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the deployment configuration object for the application and assign it to the deployment variable:
   ```
   set deployment [$AdminConfig getid /Deployment:myApp/]
   ```
   where *myApp* is the name of the application for which you want to configure session managmement.
5. Run this command to retrieve the applicaton deployment and assign it to the appDeploy variable:
   ```
   set appDeploy [$AdminConfig showAttribute $deployment deployedObject]
   ```
6. Run the attributes command to obtain a list of attributes you can set for session manager, use the attributes command:
   ```
   $AdminConfig attributes SessionManager
   ```
   The command displays this output:
   ```
   "accessSessionOnTimeout Boolean"
   "allowSerializedSessionAccess Boolean"
   "context ServiceContext@"
   "defaultCookieSettings Cookie"
   "enable Boolean"
   "enableCookies Boolean"
   "enableProtocolSwitchRewriting Boolean"
   "enableSSLTracking Boolean"
   "enableSecurityIntegration Boolean"
   "enableUrlRewriting Boolean"
   "maxWaitTime Integer"
   "properties Property(TypedProperty)*"
   "sessionDRSPersistence DRSSettings"
   "sessionDatabasePersistence SessionDatabasePersistence"
   "sessionPersistenceMode ENUM(DATABASE, DATA_REPLICATION, NONE)"
   "tuningParams TuningParams"
   ```
   Notes:
   - The sessionDRSPersistence attribute is not supported in WebSphere Application Server - Express.
   - In WebSphere Application Server - Express, the only supported value for sessionPersistenceMode is NONE.
7. Run these commands to set the attributes for the session manager for your application:
   ```
   set attr1 [list enableSecurityIntegration true]
   set attr2 [list maxWaitTime 30]
   set attr3 [list sessionPersistenceMode NONE]
   set attrs [list $attr1 $attr2 $attr3]
   set sessionMgr [list sessionManagement $attrs]
   ```
   You can modify the example to set other attributes of session manager including the nested attributes in Cookie, DRSSettings, SessionDataPersistence, and TuningParms object types. To list the attributes for those object types, use the attribute command in AdminConfig object.
8. Run this command to create the session manager for the application:
   ```
   $AdminConfig create ApplicationConfig $appDeploy [list $sessionMgr]
   ```

9. Run this command to save your changes:
```
$AdminConfig save
```

Map virtual hosts for Web modules

The Web modules installed on your application server can be mapped to different virtual hosts. You can map a virtual host for a Web module if you want that Web module to process requests from a specific URL.

To use the administrative console to map virtual hosts for Web modules, follow these steps:
1. “Start the WebSphere administrative console” on page 83.
2. Expand **Applications** and click **Enterprise Applications**.
3. Click the name of the application for which you want to manage session tracking.
4. On the application’s page, click **Map virtual hosts for Web modules**.
5. Select the Web module for which you want to map a virtual host.
6. Select the virtual host to map to the Web module.
7. **OK**.
8. “Save the application server configuration” on page 84.

Configure session management for Web modules

Application Web modules use sessions to associate HTTP requests from a single client. Session management settings determine how your application manages HTTP sessions.

You can configure sessions for a Web module if you want the module to use different settings from other Web modules that are deployed on your application server, including other Web modules in the same application. The Web module settings take precedence over the application and application server settings.
- Use the administrative console to configure session management for Web modules (page 70)
- Use wsadmin to configure session management for Web modules (page 70)

Use the administrative console to configure session management for Web modules
1. “Start the WebSphere administrative console” on page 83.
2. Expand **Applications** and click **Enterprise Applications**.
3. Click the name of the application for which you want to manage session tracking.
4. On the application’s page, click **Web Modules**.
5. Click the URI for the Web module for which you want to configure session tracking.
6. On the Web module’s page, click **Session Management**.
7. Configure session management settings.
8. Click **Apply** or **OK**.
9. “Save the application server configuration” on page 84.

Use wsadmin to configure session management for Web modules
1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the wsadmin tool:
```
cd /QIBM/ProdData/WebASE/ASE5/bin
```
3. Start wsadmin (page 97).
4. At the wsadmin prompt, run this command to identify the deployment configuration object for the application and assign it to the deployment variable:
```
set deployment [$AdminConfig getid /Deployment:myApp/]
```
where *myApp* is the name of the application for which you want to configure session management.

5. Run these commands to list all of the modules in the application and assign them to the modules variable:

```
set appDeploy [$AdminConfig showAttribute $deployment deployedObject]
set modules [lindex [$AdminConfig showAttribute $appDeploy modules] 0]
```

6. Use the attributes command to display a list of the attributes that you can set for the session manager:

```
$AdminConfig attributes SessionManager
```

The command displays this output:

```
"accessSessionOnTimeout Boolean"
"allowSerializedSessionAccess Boolean"
"context ServiceContext@"
"defaultCookieSettings Cookie"
"enable Boolean"
"enableCookies Boolean"
"enableProtocolSwitchRewriting Boolean"
"enableSSLTracking Boolean"
"enableSecurityIntegration Boolean"
"enableUrlRewriting Boolean"
"maxWaitTime Integer"
"properties Property(TypedProperty)*"
"sessionDRSPersistence DRSSettings"
"sessionDatabasePersistence SessionDatabasePersistence"
"sessionPersistenceMode ENUM(DATABASE, DATA_REPLICATION, NONE)"
"tuningParams TuningParams"
```

Notes:

- The sessionDRSPersistence attribute is not supported in WebSphere Application Server - Express.
- In WebSphere Application Server - Express, the only supported value for sessionPersistenceMode is NONE.

7. Set attributes for the session manager. For example, run these commands to set the `enableSecurityIntegration`, `maxWaitTime`, `sessionPersistenceMode`, and `enable` attributes

```
set attr1 [list enableSecurityIntegration true]
set attr2 [list maxWaitTime 30]
set attr3 [list sessionPersistenceMode NONE]
set attr4 [list enable true]
set attrs [list $attr1 $attr2 $attr3 $attr4]
set sessionMgr [list sessionManagement $attrs]
```

You can set attributes in the session manager, including the nested attributes in Cookie, DRSSettings, SessionDataPersistence, and TuningParms object types. To list the attributes for those object types, run the attribute command for each object type.

8. Run these commands to set the attributes for Web module:

```
set nameAttr [list name myWebModuleConfig]
set descAttr [list description "myWebModuleDescription"]
set webAttrs [list $nameAttr $descAttr $sessionMgr]
```

where *myWebModuleConfig* is the name for the configuration and *myWebModuleDescription* is a brief description of the configuration.

9. Create the session manager for each Web module in the application:

```
foreach module $modules {
    if {[regexp WebModuleDeployment $module] == 1} {
      $AdminConfig create WebModuleConfig $module $webAttrs
    }
  }
```

You can modify this example to set other attributes of session manager in Web module configuration.

10. Run this command to save your changes:

```
$AdminConfig save
```

Use the EARExpander script to work with applications

The EARExpander script expands an EAR file into a directory to run the application in that EAR file. It also collapses a directory containing application files into a single EAR file. You can type EARExpander with no arguments to learn more about its options. For information on running Qshell scripts, see "Configure Qshell to run WebSphere Application Server - Express scripts" on page 92.

Authority

No special authority is required to run this script.

Usage

To run the script, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the EARExpander script:
   ```
   EARExpander -ear earfile -operation expand | collapse
   -operationDir directory
   ```
   where *earfile* is the name of the EAR file that you want to expand or the name of the EAR file that you want to create and *directory* is the name of the directory to which the EAR file is expanded or from which the EAR file is created.

Syntax

The syntax of the EARExpander script is:

```
EARExpander -ear earfile -operation expand | collapse
  -operationDir directory [ -help | -? ]
```

Parameters

The parameters of the EARExpander script are:

- **-ear**
  This is a required parameter. The value *earfile* specifies the name of the EAR file that the expand or collapse operation uses. It is recommended that you specify a fully qualified name for the EAR file.
  - If you specify the expand operation, this EAR file is extracted to the directory specified by the -operationDir parameter. If you do not specify the fully qualified name of the EAR file, the script searches the current working directory for the EAR file.
  - If you specify the collapse operation, the script creates this EAR file from the contents of the directory specified by the -operationDir parameter. For a collapse operation, you must specify a new name for the EAR file. If you do not specify the fully qualified name of the EAR file, the script creates the EAR file in the current working directory.
- **-operation**
  This is a required parameter. Valid arguments are expand and collapse.
  - If you specify expand, the script extracts the contents of the EAR file into the directory specified by -operationDir.
  - If you specify collapse, the script creates an EAR file from the contents of directory specified by -operationDir.
- **-operationDir**
  This is a required parameter. The value *directory* specifies the directory on which the expand or collapse operation runs.
  - If you specify the expand operation, the EAR file is expanded to this directory. If the directory does not exist, the script creates it.

  – If you specify the collapse operation, the EAR file is created from the contents of this directory.
- **-help** or **-?**
  This optional paramter prints the usage statement for the script.

Examples

```
EARExpander -ear /home/myProfile/myFile.EAR -operation expand
 -operationDir /home/myProfile/myApps/myApp1
```

This example expands the file myFile.EAR into the /home/myProfile/myApps/myApp1 directory.

```
EARExpander -ear /home/myProfile/myNewFile.EAR -operation collapse
 -operationDir /home/myProfile/myApps/myNewApplication
```

This example creates the file myNewFile.EAR from the contents of the /home/myProfile/myApps/myNewApplication directory.

Backup and recovery considerations for WebSphere Application Server - Express

WebSphere Application Server - Express uses many iSeries resources that you should consider adding to your backup and recovery procedures. Save and restore of WebSphere Application Server - Express resources uses the same iSeries commands used for other iSeries resources. For more detailed information on iSeries system save and restore facilities, see the following topics in the iSeries Information Center:
- Backup and Recovery
- Availability

The following areas of WebSphere Application Server - Express should be considered for backup and recovery:

“Backup and recovery: Administrative configuration” on page 74
This topic describes the backup and recovery of the configuration for a WebSphere Application Server - Express instance.

“Backup and recovery: servlets” on page 75
This topic describes the backup and recovery of the collections and files used by the servlet function of WebSphere.

“Backup and recovery: JavaServer Pages (JSP) files” on page 76
This topic describes the backup and recovery of the resources needed to configure and run JavaServer Pages (JSP) files.

“Backup and recovery: Security” on page 76
This topic describes considerations for saving and restoring security configuration and data.

Other external resources

If your applications are using other resources or services that are external to WebSphere Application Server - Express remember to include those in your backup plan as well.

WebSphere Application Server - Express resources can be saved while the WebSphere Application Server - Express environment is active. When backing up database data, you may have to shut down some or all services if a snapshot cannot be obtained. This would occur if there are requests which obtain locks or have open transactions against the database being saved. In a distributed environment, you may need to consider how to get a consistent backup across several systems. If the data on systems is not closely related to data on other systems, you may be able to backup each system in isolation. If you need a snapshot across systems simultaneously, you may need to stop activity on all systems while the snapshot is taken.

How often you back up resources depends largely on when or how often you expect them to change. Use these categories to determine how you should fit WebSphere resources into your backup plan:

- **WebSphere Application Server - Express environment configuration**
  This category includes the resources that define your WebSphere Application Server - Express operating environment. After you install and configure the product, this information should change very infrequently. You might backup this information only when you change these settings, and not include these resources in regularly scheduled backups. This category includes these items:
  - "Backup and recovery: Administrative configuration"
  - "Backup and recovery: servlets" on page 75
  - "Backup and recovery: Security" on page 76
- **WebSphere Application Server - Express applications**
  This category includes the applications that you run in WebSphere Application Server - Express. You should back these up the same way you back up other applications on your system. You could back up these resources every time you add or change an application, or include these resources in a regularly scheduled backup. This category includes these items:
  - "Backup and recovery: Administrative configuration"
  - "Backup and recovery: servlets" on page 75
  - "Backup and recovery: JavaServer Pages (JSP) files" on page 76
- **WebSphere Application Server - Express application data**
  This category includes the data stores used by your WebSphere Application Server - Express applications. Unless your applications serve only static information, these resources are usually quite dynamic. You should back these up the same way you back up other business data on your system. These resources are suited for inclusion in a regularly scheduled backup.

Backup and recovery: Administrative configuration

The administrative configuration contains vital information regarding your WebSphere Application Server - Express setup, and it should be backed up.

Administration server configuration

Most of the configuration for your WebSphere Application Server - Express instance resides in the config directory structure. In addition, the properties directory also contains several important configuration files. For more information on the configuration content stored in the properties directory, see "Properties files" on page 142. For more information about the configuration content stored in the config directory structure, see "Administrative repository" on page 139.

To save these properties files, run the Save (SAV) command:

```
SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
 OBJ(('/QIBM/UserData/WebASE/ASE5/instance/properties/*')
 ('/QIBM/UserData/WebASE/ASE5/instance/config/*'))
```

where *instance* is the name of your instance.

Note: This command has been wrapped for display purposes. Enter it as one command.

To restore the properties files, run the Restore (RST) command:

```
RST DEV('/QSYS.lib/wsalib.lib/wsasavf.file') OBJ('/QIBM/*')
```

Backup and recovery: servlets

The following items should be considered for backup when using servlets:

- servlet source and class files
- HTTP configuration
- "Backup and recovery: Administrative configuration" on page 74

Servlet source and class files

Application code and configuration (such as bindings) is located by default in the /QIBM/UserData/WebASE/ASE5/*instance*/installedApps directory. By saving this directory, you save your installed applications, including HTML, servlets, and JavaServer Pages (JSP) files. Normally, each application is located in a separate subdirectory, so you can choose to save all applications or a subset.

Note: The commands below have been wrapped for display purposes. Enter each as a single command.

This command saves all installed applications:

```
SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
 OBJ(('/QIBM/UserData/WebASE/ASE5/instance/installedApps'))
```

This command saves the sampleApp application only:

```
SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
 OBJ(('/QIBM/UserData/WebASE/ASE5/instance/installedApps/cellName/sampleApp.ear'))
```

If you have located utility or general purpose classes in other directories, such as /QIBM/UserData/WebASE/ASE5/*instance*/lib/app or /QIBM/UserData/WebASE/ASE5/*instance*/lib/ext, be sure to include those locations in your backup plan as well.

HTTP configuration

Note: The following information applies to IBM HTTP Server for i5/OS (powered by Apache). If you are using Lotus Domino HTTP Server, see the Notes.net Documentation Library (http://www.notes.net/notesua.nsf?OpenDatabase)

.

Changes to the HTTP configuration are often made to enable WebSphere Application Server - Express to serve servlets and JSP requests, and to enable WebSphere Application Server - Express security. You should consider saving your HTTP configuration as a part of your WebSphere Application Server - Express backup and recovery.

HTTP server instances for IBM HTTP Server for i5/OS (powered by Apache) are members of the QATMHINSTC file in the library QUSRSYS. This is an example save command for this file:

```
SAVOBJ OBJ(QATMHINSTC) LIB(QUSRSYS) DEV(*SAVF) OBJTYPE(*FILE) SAVF(WSALIB/WSASAVF)
```

The HTTP configurations for IBM HTTP Server for i5/OS (powered by Apache) are stored in the integrated file system in a subdirectory, chosen when the configuration was created. The recommended location is within the WebSphere instance directory. You can determine this file location by inspecting HTTP server instance member in the QATMHINSTC file in library QUSRSYS. This is an example save command for this file:

```
SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file') OBJ(('/QIBM/UserData/WebASE/ASE5/instance/apache/conf')
  ('/QIBM/UserData/WebASE/ASE5/instance/htdocs'))
```

where *instance* is the name of your instance.

Backup and recovery: JavaServer Pages (JSP) files

The following items should be considered for backup when using JavaServer Pages(TM) (JSP) files:

- JSP source and generated servlet classes
- HTTP configuration
- "Backup and recovery: Administrative configuration" on page 74

JSP source and generated servlet classes

Application code and configuration (such as bindings) is located by default in the /QIBM/UserData/WebASE/ASE5/*instance*/installedApps directory. By saving this directory, you save your installed applications, including HTML, servlets, and JavaServer Pages (JSP) files. Normally, each application is located in a separate subdirectory, so you can choose to save all applications or a subset.

Note: The following commands have been wrapped for display purposes. Enter each as a single command.

This command saves all installed applications:

```
SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
 OBJ(('/QIBM/UserData/WebASE/ASE5/instance/installedApps'))
```

This command saves the sampleApp application only:

```
SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
 OBJ(('/QIBM/UserData/WebASE/ASE5/instance/installedApps/cellName/sampleApp.ear'))
```

When JSP files run, a servlet class is generated, compiled, and then run. When saving and restoring your JSP files, you can elect to save only the JSP source or the generated files as well.

- If you save and restore only the JSP source, the servlet source and class files are regenerated when they are invoked. This is a simpler, smaller save and restore operation. Note that regeneration slows the first requests, and default optimization is done on the generated Java programs.
- If you save and restore the source and generated files, no regeneration is done. If you have optimized Java programs to levels other than the default, this optimization is preserved.

WebSphere Application Server - Express places the generated files (.class, .java, and optionally, .dat) in a temporary directory under the WebSphere Application Server - Express instance. For example:

```
/QIBM/UserData/WebASE/ASE5/instance/temp/node/
appserver/application/web_module
```

where *instance* is the name of your instance, *node* is the name of the iSeries server or partition on which your WebSphere Application Server - Express instance is running, *appserver* is the name of your application server instance, *application* is the name of the enterprise application to which the JSP file belongs, and *web_module* is the Web module that contains your JSP file.

Note: A .dat file is a helper file used by the generated servlet.

Backup and recovery: Security

These items should be considered for backup of security information:

- Users
- Security properties files
- "Backup and recovery: Administrative configuration" on page 74
- HTTP configuration

- Key files
- Validation lists

Users

When using local OS security, back up your i5/OS user profiles, using normal i5/OS save procedures for user profiles. For more information, see the following topics in the iSeries Information Center:
- Backup and Recovery
- Availability
- Backup and Recovery

For information on the Directory Server Product (LDAP server), see the IBM Directory Server for iSeries (LDAP) topic in the iSeries Information Center.

For information on Domino, see the Domino Reference Library (http://doc.notes.net/domino_notes/5.0/as400/as400hlp.nsf)

.

Security properties files

Security settings are saved in several properties files. By default, these are located in /QIBM/Userdata/WebASE/ASE5/*instance*/properties where *instance* is the name of your instance. If you have defined additional WebSphere instances, you will have additional properties files located in the directories for those instances.

```
SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
 OBJ(('instance/properties/sas*'))
```

Note: This command has been wrapped for display purposes. Enter it as one command.

Security property files can be saved while WebSphere is running.

HTTP configuration

Note: The following information applies to IBM HTTP Server for i5/OS. If you are using Lotus Domino HTTP Server, see the Notes.net Documentation Library (http://www.notes.net/notesua.nsf?OpenDatabase)

.

Changes to the HTTP configuration are often made to enable WebSphere Application Server - Express to serve servlets and JSP requests, and to enable WebSphere Application Server - Express security. You should consider saving your HTTP configuration as a part of your WebSphere Application Server - Express backup and recovery. The IBM HTTP Server configurations are stored as members of the QATMHTTPC file in library QUSRSYS. HTTP server instances are members of the QATMHINSTC file in the library QUSRSYS. These are example save commands for these files:

```
SAVOBJ OBJ(QUSRSYS/QATMHTTPC)
SAVOBJ OBJ(QUSRSYS/QATMHINSTC)
```

Key files

Key files should also be saved. They contain certificates used by the WebSphere Application Server - Express security infrastructure and also for HTTPS transport between servers. Save all files in the WAS_INSTANCE_ROOT/etc directory. Key files are contained in the WAS_INSTANCE_ROOT/etc directory, but may be created and stored in other directories by administrators.

Validation lists

Passwords are stored as encrypted data in validation list objects when the i5/OS password encoding algorithm is used. The default validation list is /QSYS.LIB/QUSRSYS.LIB/EJSADMIN.VLDL, but you can change it in the WebSphere administrative console by specifying it as a system property for the application server.

Save and restore validation list objects using the Save Object (SAVOBJ) and Restore Object (RSTOBJ) commands, for example:

```
SAVOBJ OBJ(EJSADMIN) LIB(QUSRSYS) DEV(*SAVF) SAVF(WSALIB/WSASAVF)
 RSTOBJ OBJ(EJSADMIN) SAVLIB(QUSRSYS) DEV(*SAVF) OBJTYPE(*VLDL) SAVF(WSALIB/WSASAVF)
```

Back up the application server configuration

To back up the configuration of your application server, run the backupConfig script in Qshell. The backupConfig script backs up the configuration of your application server to a ZIP file. By default, the application server stops before the backup is made so that partially synchronized information is not saved. For information on running Qshell scripts, see “Configure Qshell to run WebSphere Application Server - Express scripts” on page 92.

Note: The backupConfig script only backs up information for a single instance of WebSphere Application Server - Express. For a more complete backup configuration, it is recommended that you follow the procedures described in “Backup and recovery considerations for WebSphere Application Server - Express” on page 73.

Authority

To run this script, your user profile must have *ALLOBJ authority.

Usage

To run the script, follow these steps:

1. On an CL command line, enter the Start Qshell (STRQSH) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the backupConfig script:
   ```
   backupConfig -instance instance
   ```
   where *instance* is the name of your application server instance.

Syntax

The syntax of the backupConfig script is:

```
backupConfig -instance instance [ backup_file ]
 [ -nostop ] [ -nowait ] [ -quiet ] [ -logfile filename ]
 [ -replacelog ] [ -trace ] [ -username username ]
 [ -password password ] [ -help | -? ]
```

Parameters

The parameters of the backupConfig script are:

- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance for which you want to back up the configuration.
- ***backup_file***
  This optional parameter specifies the fully qualified file name to which the script writes the backup configuration. If you do not specify this parameter, the script creates a file named WebSphereConfig_*timestamp*.zip in the current working directory, where *timestamp* is the current date in YYYY-MM-DD format.

  Note: The QEJBSVR profile must have *RWX access to the directory where the backup file is created.
- **-nostop**
  This is an optional parameter. If you specify this parameter, the backupConfig script does not to stop the servers before it creates the backup configuration. By default, the script stops the servers before it creates the backup configuration.
- **-nowait**
  This is an optional parameter. If you specify this parameter, the script returns control to the user without waiting for the backup configuration to be created. The default is to wait for the backup configuration to be created.
- **-quiet**
  This is an optional parameter. If you specify this parameter, the script does not display informational messages. The default is to display informational messages while the script runs.
- **-logfile**
  This is an optional parameter. The value *filename* specifies the location and name of the log file for the script. The default value is /QIBM/UserData/WebASE/ASE5/*instance*/logs/backupConfig.log, where *instance* is the name of the instance for which you want to back up the configuration.
- **-replacelog**
  This is an optional parameter. If you specify this parameter, the script replaces the log file if it exists. By default the script appends to the log file if it exists.
- **-trace**
  This is an optional parameter. If you specify this parameter, the script outputs additional trace information to the log file for the script. You should only specify this parameter if errors occur when you try to back up a configuration. The default is to not log additional trace information.
- **-username**
  This parameter is required if security is enabled for the server. The value *username* specifies the user name for authentication. The script uses this parameter to stop the servers. If you specify the -nostop parameter, you do not need to specify the -username parameter.
- **-password**
  This parameter is required if security is enabled for the server. The value *password* specifies the password for authentication. The script uses this parameter to stop the servers. If you specify the -nostop parameter, you do not need to specify the -password parameter.
- **-help** or **-?**
  This optional paramter prints the usage statement for the script.

Example

```
backupConfig -instance devinst /QIBM/UserData/WebASE/ASE5/devinst/devConfig.zip
```

This example backs up the configuration for the devinst instance. The script creates a backup configuration file called devConfig.zip in the /QIBM/UserData/WebASE/ASE5/devinst directory.

Restore the application server configuration

The restoreConfig script restores the configuration of a node. To run this script, you must have a backup copy of the configuration. If the configuration directory already exists, the script renames it before restoring the configuration. For information on running Qshell scripts, see “Configure Qshell to run WebSphere Application Server - Express scripts” on page 92.

Note: The restoreConfig script only restores information for a single instance of WebSphere Application Server - Express. For a more complete restoration of your configuration, it is recommended that you follow the procedures described in “Backup and recovery considerations for WebSphere Application Server - Express” on page 73.

Authority

To run this script, your user profile must have *ALLOBJ authority.

Usage

To restore your application server configuration, follow these steps:

1. On the CL command line, run the STRQSH (Start Qshell) command.
2. Run the cd command to change to the directory that contains the script:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. Run the restoreConfig script:
   ```
   restoreConfig backup_file -instance instance
   ```
   where *backup_file* is the name of the configuration file that you created with the backupConfig script and *instance* is the name of the instance to which you want to restore the configuration.

Syntax

The syntax of the restoreConfig script is:

```
restoreConfig backup_file -instance instance [ -nostop ] [ -quiet ]
 [ -logfile filename ] [ -replacelog ] [ -trace ] [ -username username ]
 [ -password password ] [ -help | -? ]
```

Parameters

The parameters of the restoreConfig script are:

- ***backup_file***
  This is a required parameter. The value *backup_file* specifies the fully qualified name of the backup configuration that you want to restore. This file is generated when you run the backupConfig script.
- **-instance**
  This is a required parameter. The value *instance* specifies the name of the instance to which you want to restore the configuration.
- **-nostop**
  This is an optional parameter. If you specify this parameter, the restoreConfig script does not to stop the servers before it restores the configuration. By default, the script stops the servers before it restores the configuration.
- **-nowait**
  This is an optional parameter. If you specify this parameter, the script returns control to the user without waiting for the configuration to be restored. The default is to wait for the configuration to be restored.
- **-quiet**
  This is an optional parameter. If you specify this parameter, the script does not display informational messages. The default is to display informational messages while the script runs.
- **-logfile**
  This is an optional parameter. The value *filename* specifies the location and name of the log file for the script. The default value is /QIBM/UserData/WebASE/ASE5/*instance*/logs/restoreConfig.log, where *instance* is the name of the instance for which you want to restore the configuration.

- **-replacelog**
  This is an optional parameter. If you specify this parameter, the script replaces the log file if it exists. By default the script appends to the log file if it exists.
- **-trace**
  This is an optional parameter. If you specify this parameter, the script outputs additional trace information to the log file for the script. You should only specify this parameter if errors occur when you try to restore a configuration. The default is to not log additional trace information.
- **-username**
  This parameter is required if security is enabled for the server. The value *username* specifies the user name for authentication. The script uses this parameter to stop the servers. If you specify the -nostop parameter, you do not need to specify the -username parameter.
- **-password**
  This parameter is required if security is enabled for the server. The value *password* specifies the password for authentication. The script uses this parameter to stop the servers. If you specify the -nostop parameter, you do not need to specify the -password parameter.
- **-help** or **-?**
  This optional paramter prints the usage statement for the script.

Example

```
restoreConfig -instance myAppSvr /QIBM/UserData/WebASE/ASE5/myAppSvr/myConfig.zip
```

This example restores the configuration for the myAppSvr instance. The script uses the configuration file called myConfig.zip in the /QIBM/UserData/WebASE/ASE5/myAppSvr directory.

Reference

These topics provide reference information on WebSphere Application Server - Express and the administrative tools provided with the product.

"Administrative tools" on page 82
This topic provides a brief overview of the tools that you can use to configure and manage your application server and applications. It includes information to help you select a tool to use for administrative tasks.

"The WebSphere administrative console" on page 83
This topic describes the WebSphere administrative console and how to use it.

"Qshell scripts" on page 90
WebSphere Application Server - Express includes several administrative Qshell scripts and commands. This topic provides a description of each script and how to use it.

"The wsadmin administrative tool" on page 96
This topic describes how to use wsadmin, the command line administrative client provided with WebSphere Application Server - Express. The wsadmin tool allows you to perform administrative functions from the Qshell command line, interactively or by providing scripts as input.

"The ws_ant script" on page 130
Apache Ant is a Java-based build tool. This topic provides information about ws_ant, which is a WebSphere implementation of the Ant.

"Java Management Extensions" on page 133
Java Management Extensions are a set of Java administrative APIs that you can use to customize WebSphere Application Server - Express administration. This describes how to use JMX. JMX is provided with WebSphere Application Server - Express as a technology preview.

“Product library, directories, and subsystems” on page 138
This topic describes the product library and the integrated file system directories for the product, and the subsystem for WebSphere Application Server - Express.

“Administrative repository” on page 139
Each instance of WebSphere Application Server - Express includes several configuration documents which collectively form the administrative repository for the instance. These documents describe the servers, nodes, applications, and resources that comprise the administrative domain of the instance. This topic provides information about the administrative repository and the documents it contains

“Properties files” on page 142
Each instance of WebSphere Application Server - Express includes several properties files. This topic describes the properties files included in the properties directory for each instance.

“WebSphere Application Server - Express default port definitions” on page 143
This topic describes the TCP/IP ports that WebSphere Application Server - Express uses, and how to change the default port numbers.

“User profiles and authorities” on page 145
This topic describes the user profiles that WebSphere Application Server - Express uses, and lists the necessary authorities for the user profile under which WebSphere Application Server - Express will run.

“Set the time zone” on page 145
This topic describes how to set the time zone and provides the supported values for the user.timezone property.

Administrative tools

WebSphere Application Server - Express includes these administrative tools:

- **The HTTP Server Administration interface**
  This interface is part of IBM HTTP Server for i5/OS (powered by Apache). You can use it to perform the common tasks for managing your application server instance, including starting and stopping the application server, installing and starting applications, and configuring database access for your applications. If you are performing basic administrative tasks, you do not need to use the other administrative tools. However, several advanced configuration tasks cannot be completed with the HTTP Server Administration interface.
- **The WebSphere administrative console**
  The administrative console is a browser-based interface that allows you to configure application server settings, deploy and manage applications, and perform additional tasks that are not included in the HTTP Server Administration interface. See “The WebSphere administrative console” on page 83 for information on using the administrative console.
- **Qshell scripts**
  WebSphere Application Server - Express includes several Qshell scripts that you can use to perform basic tasks such as creating an application server and starting and stopping servers. You can use these scripts to create and start application servers from a 5250 emulator session. For information on running Qshell scripts, see “Qshell scripts” on page 90.
- **The wsadmin tool**
  The wsadmin tool is an interactive scripting tool that you can use to configure and manage application servers and applications. You can use wsadmin to configure and manage application servers, applications, application server resources, and security. See “The wsadmin administrative tool” on page 96 for information on using wsadmin.

The WebSphere administrative console

The WebSphere administrative console is a browser-based graphical administrative interface for configuring and managing WebSphere resources. You can use the WebSphere administrative console to display and change your WebSphere Application Server - Express configurations, and to manage your WebSphere Application Server - Express resources.

These topics provide general information about the WebSphere administrative console:

"Start the WebSphere administrative console"
This topic describes how to start the WebSphere administrative console for WebSphere Application Server - Express.

"Save the application server configuration" on page 84
This topic describes how to save changes to your application server's configuration.

"WebSphere administrative console features" on page 84
This topic provides information on WebSphere administrative console features, such as the navigation menu, taskbar, and the **Configuration** and **Runtime** tabs, as well as how to set preferences and access help text.

"WebSphere administrative console topology reference" on page 86
This topic provides an outline of the objects in the WebSphere administrative console.

"Administer console users and groups" on page 89
If you have multiple users working with the WebSphere administrative console in a secured environment, you can manage those users through the WebSphere administrative console. This topic describes how to manage those users and groups.

The WebSphere administrative console includes information about using the console, as well as detailed help text about its features and preference settings. It also provides descriptions of the properties that you can edit with the console. To access this information, click **Help** in the WebSphere administrative console's taskbar.

Start the WebSphere administrative console

To start the administrative console on a workstation, follow these steps:

1. Start the HTTP Server Administration interface.
2. Click the **Manage** tab.
3. Select your application server from the **Server** list.
4. Enable the adminconsole application if it is not already enabled.
   a. In the navigation menu, click **Manage installed applications**.
   b. Select the adminconsole application.
   c. Click **Properties**.
   d. For **Application Enablement**, select **Enabled**.
5. To start your application server, click the Start button (

) at the top of navigation menu. If your application server is running, you must stop it and restart it.
6. In the navigation menu, click **Launch Express Console**.
7. When prompted, enter a user ID. The WebSphere administrative console is displayed in the browser window.

Note: The user ID does not need to be an i5/OS user profile. This user ID is used only to track which users make changes to the application server configuration.

You can also start the console without starting the HTTP Server Administration interface. To use this method, follow these steps:

1. Start the HTTP Server Administration interface.
2. Click the **Manage** tab.
3. Select your application server from the **Server** list.
4. Enable the adminconsole application if it is not already enabled.
   a. In the navigation menu, click **Manage installed applications**.
   b. Select the adminconsole application.
   c. Click **Properties**.
   d. For **Application Enablement**, select **Enabled**.
5. To start your application server, click the Start button ( ) at the top of navigation menu. If your application server is running, you must stop it and restart it.
6. Open this URL in your browser:

   `http://`*`your.server.name`*`:`*`port`*`/admin`

   where *your.server.name* is the host name of the iSeries server on which your application server is running, and *port* is the WebSphere administrative console port as noted in the ready message in the joblog of your application server. This port is also specified in the admin_host virtual host.
7. When prompted, enter a user ID. The WebSphere administrative console is displayed in the browser window.

   Note: The user ID does not need to be an i5/OS user profile. This user ID is used only to track which users make changes to the application server configuration.

Save the application server configuration

When you make changes to your application server's configuration, you must save the changes before they take effect. The WebSphere administrative console prompts you with a message at the top of the browser window when it detects that changes have been made.

To save the updated application server configuration, follow these steps:

1. Click **Save** in the message or in the task menu.
2. On the **Save to Master Configuration** screen, you can save or discard your changes.
   - To save your changes, click **Save**.
   - To discard your changes, click **Discard**, then click **Yes**.

WebSphere administrative console features

This topic provides information about features of the WebSphere administrative console. These features can help you navigate the console and perform the tasks described in this documentation.

- Areas of the console (page 84)
- Preferences settings (page 85)
- Page features (page 85)

Areas of the console

These are the main areas of the administrative console:

- **Taskbar**
  The taskbar is near the top of the browser window of the administrative console. Use the links on the taskbar to perform these basic tasks:
  – Display the administrative console home page.
  – Save changes to the administrative configuration.
  – Set console preferences.
  – Log out of your administrative console session.
  – Access help topics for the console.
- **Navigation tree**
  The navigation tree on the left side of the console contains links to pages that you use to create and manage components of your WebSphere Application Server - Express application server instance.
- **Workspace**
  The workspace is on the right side of the console. This area displays the pages that you use to create and manage components of your WebSphere Application Server - Express application server instance.
- **WebSphere Status**
  The status messages area at the bottom of the console displays error messages and runtime event messages.

Preferences settings

Use the **Preferences** page to configure these settings for the administrative console:
- Enable disable workspace auto-refresh
- Confirm workspace discards
- Use the default scope
- Hide or show administrative console banner
- Hide or show page and field descriptions

Page features

The workspace area of the console displays collection pages, detail pages, and wizard pages. Administrative console pages include these features:
- **Collection pages**
  A collection page displays a set of administrative objects.
  – Use the scope settings to filter the contents of an administrative console collection table to a particular cell, node, or server.
    - Cell limits the visibility to all servers on the named cell.
    - Node limits the visibility to all servers on the named node.
    - Server limits the visibility to the named server.

    The server scope has precedence over the node and cell scopes, and the node scope has precedence over the cell scope. Note that objects are created at only one scope, though they might be visible at more than one scope.
  – Use the filter settings to display a subset of the collection.
  – The page preferences settings determine how information is displayed on the collection page.
  – The table of objects lists the object names, and may list one or more additional properties of the objects.
  – The page includes buttons that you use to perform actions on the objects in the collection. For example, you can create a new object, delete an object, or start and stop an object.
  – Each column heading includes buttons to sort the contents of the table in ascending or descending order by a column.

- **Detail pages**
  Use a detail page to configure a specific administrative object, such as an application server.
  - **Configuration tabbed page**
    Use this tabbed page to view and modify the configuration of an administrative object.
  - **Runtime tabbed page**
    Use this tabbed page to view the configuration that is currently in use for the administrative object. For most objects, this page is read-only.
  - Detail pages include buttons that you use to work with the object's configuration.
- **Wizard pages**
  Use wizard pages to perform a configuration process that includes several steps. A wizard performs a task based on information that you specify.

WebSphere administrative console topology reference

This outline shows the topology of objects in the WebSphere administrative console. The cell name is *host_instance*, where *host* is the name of your iSeries server, and *instance* is the name of your application server instance. For example, if the name of your machine is MYISERIES, and you create an instance named myAppSvr, the cell name is MYISERIES_myAppSvr.

- *Cell Name*
  - Servers
    - Application Servers
      - *Application Server Name*
        - Web Container
          - Thread Pool
          - Session Management
            - Cookies
          - HTTP Transports
            - *Host Name*
              - Custom Properties
              - SSL Configuration Repertoire - Cell Level
          - Custom Properties
        - Logging and Tracing
          - Diagnostic Trace
          - JVM Logs
          - Process Logs
          - IBM Service Logs
        - ORB Service
          - Thread Pool
          - Custom Properties
        - Custom Properties
        - Diagnostic Trace Service
        - Debugging Service
        - IBM Service Logs
        - Server Components
          - Name Server
            - Custom Properties
          - Application Server
        - Process Definition

          - Java Virtual Machine
            - Custom Properties
          - Process Execution
          - Process Logs
          - Environment Entries
          - Monitoring Policy
        - Performance Monitoring Service
          - Custom Properties
        - End Points
          - *End Point Name*
        - Classloader
          - *Classloader ID*
- Applications
  - Enterprise Applications
    - *Application Name*
      - Target Mappings
        - *Target Name*
      - Libraries
      - Session Management
        - Cookies
      - View Deployment Descriptor
      - Map security roles to users/groups
      - Map virtual hosts for web modules
      - Web Modules
        - *URI*
          - Target Mappings
          - Session Management
          - View Deployment Descriptor
      - Connector Modules
  - Install New Application
- Resources
  - JDBC Providers
    - *JDBC Provider Name*
      - Data Sources
        - *Data Source Name*
          - Connection Pool
          - Custom Properties
          - J2C Authentication Data Entries
            - *Alias*
      - Data Sources (Version 4)
        - *Data Source Name*
          - Connection Pool
          - Custom Properties
  - Mail Providers
    - *Mail Provider Name*

        – Protocol Providers
            - *Protocol Name*
        – Mail Sessions
            - *Mail Session Name*
                • Custom Properties
        – Custom Properties
- Resource Adapters
    • *Resource Adapter Name*
        – CMP Connection Factories
        – Custom Properties
        – View Deployment Descriptor

– Security
- Global Security
    • Custom Properties
- SSL
    • *Alias*
        – Custom Properties
- Authentication Mechanisms
    • LTPA
        – Trust Association
            - Interceptors
                • *Interceptor classname*
                    – Custom Properties
        – Single Signon (SSO)
- User Registries
    • Local OS
        – Custom Properties
    • LDAP
        – Advanced LDAP Settings
        – Custom Properties
    • Custom
        – Custom Properties
- JAAS Configuration
    • Application Logins
        – *Alias*
            - JAAS Login Modules
                • *Module Classname*
                    – Custom Properties
    • J2C Authentication Data
        – *Alias*
- Authentication Protocol
    • CSIv2 Inbound Authentication
    • CSIv2 Outbound Authentication
    • CSIv2 Inbound Transport
    • CSIv2 Outbound Transport

        - SAS Inbound Transport
        - SAS Outbound Transport
  - Environment
    - Update Web Server Plugin
    - Virtual Hosts
      - *Virtual Host Name*
        - Host Aliases
          - *Host Name*
        - MIME Types
          - *MIME Type*
    - Manage WebSphere Variables
      - *Variable Name*
  - System Administration
    - Console Users
      - *User ID*
    - Console Groups
      - *Group Name*
  - Troubleshooting
    - Logs and Trace
      - *Server Name*
        - Diagnostic Trace
        - JVM Logs
        - Process Logs
        - IBM Service Logs
    - Configuration Problems

Administer console users and groups

You can use the WebSphere administrative console to give users specific authority to administer WebSphere Application Server - Express using tools such as the WebSphere administrative console or wsadmin scripting. The authority checks are only performed when global security is enabled.

For information on administering console users and groups, see these sections:
- Add console users and groups (page 89)
- Modify console users and groups (page 89)
- Remove console users and groups (page 90)
- Log out a console user (page 90)

Add console users and groups

To add a console user or group, follow these steps:
1. "Start the WebSphere administrative console" on page 83.
2. In the topology tree, expand **System Administration** and click **Console Users** or **Console Groups**.
3. On the **Console Users** or **Console Groups** page, click **Add**.
4. Specify a name from the active user registry and one or more roles for the user or group.
5. Click **OK**.
6. "Save the application server configuration" on page 84.

Modify console users and groups

To modify a console user or group, follow these steps:

1. “Start the WebSphere administrative console” on page 83.
2. In the topology tree, expand **System Administration** and click **Console Users** or **Console Groups**.
3. On the **Console Users** or **Console Groups** page, click the name of the user or group that you want to modify.
4. Make your changes.
5. Click **OK**.
6. “Save the application server configuration” on page 84.

Remove console users and groups

To remove a console user or group, follow these steps:

1. “Start the WebSphere administrative console” on page 83.
2. In the topology tree, expand **System Administration** and click **Console Users** or **Console Groups**.
3. On the **Console Users** or **Console Groups** page, select the checkbox for the user or group that you want to remove.
4. Click **Remove**.
5. “Save the application server configuration” on page 84.

Log out console users

To log out a console user, follow these steps:

1. “Start the WebSphere administrative console” on page 83.
2. In the topology tree, expand **System Administration** and click **Console Users**.
3. On the **Console Users** page, select the checkbox for the user that you want to log out.
4. Click **Logout**.

Qshell scripts

WebSphere Application Server - Express provides several scripts that you can use to administer your application server environment. These topics provide more information about the administrative Qshell scripts and how to run them.

“Configure Qshell to run WebSphere Application Server - Express scripts” on page 92
Most of the WebSphere Application Server - Express scripts are located in the /QIBM/ProdData/WebASE/ASE5/bin directory. This topic describes the different methods that you can use to invoke the Qshell scripts.

“Qshell environment variables” on page 93
This topic describes how to set environment variables that affect Qshell scripts.

“Security and Qshell scripts” on page 94
In a secured environment, you must provide authentication information for some scripts. This topic describes different ways to provide authentication information.

“Set explicit authorities for the startServer and stopServer scripts” on page 94
You can grant explicit authorities so that user profiles that do not have *ALLOBJ authority can run the startServer and stopServer scripts. This topic provides information on granting the required authorities.

Note: Parameter values that specify a server name, a node name or a cell name are case sensitive. For example, if you want to start the application server myAppServer for the instance test, invoke startServer myAppServer -instance test. If you specify myappserver for the server name, the startServer script fails.

Instance names and iSeries host names are not case sensitive.

- Application server scripts (page 91)
- Application scripts (page 91)
- Configuration scripts (page 91)
- Logging and tracing scripts (page 91)
- Serviceability tools (page 92)
- Application development scripts (page 92)
- Web services scripts (page 92)
- Security scripts (page 92)

Application server scripts

Use these scripts to manage instances of WebSphere Application Server - Express:

- "Configure a remote HTTP topology" on page 43 script creates an HTTP server instance on a machine that is not running WebSphere Application Server - Express. Your application server connects to this HTTP server to receive client requests.
- "Create an application server instance with the crtwasinst script" on page 4 script creates a new instance of WebSphere Application Server - Express.
- "Delete an application server instance with the dltwasinst script" on page 7 script deletes an instance of WebSphere Application Server - Express.
- "Display application server properties with the dspwasinst script" on page 7 script displays information about an instance.
- "Start an application server with the startServer script" on page 10 script starts an application server.
- "Stop an application server with the stopServer script" on page 13 script stops an application server.
- "Display the status of your application server with the serverStatus script" on page 14 script displays the status of an application server.
- "Change application server ports with the chgwassvr script" on page 50 script changes properties of an application server.
- "Grant authority to an instance" on page 45 grants users authority to an instance.
- "Revoke authority to an instance" on page 47 revokes a user's authority to an instance.

Application scripts

Use this script to manage applications:

- "Use the EARExpander script to work with applications" on page 72 script expands EAR files into a directory and collapses directories of application files into EAR files.

Configuration scripts

Use these scripts to manage your server's configuration:

- "Back up the application server configuration" on page 78 script backs up the configuration of a node into a ZIP file.
- "Restore the application server configuration" on page 79 script restores a configuration of a node.
- "Regenerate the Web server plugin configuration" on page 41 script regenerates the configuration file for the HTTP server plugin.

Logging and tracing scripts

These scripts provide logging and tracing information:

- Product version report scripts generate version reports on the WebSphere Application Server - Express product.
- Product history report scripts generate history reports of the WebSphere Application Server - Express product.
- The collector script script gathers information about your WebSphere Application Server - Express installation and packages it in a JAR file.
- The showlog script script displays the contents of the IBM Service log for your WebSphere Application Server - Express installation.

Serviceability tools

- The checkprereqs script invokes the prerequisite validator.
- The servicetools script invokes serviceability tools that you can use to help troubleshoot WebSphere Application Server - Express.
- The port validator verifies your WebSphere Application Server - Express configuration to ensure that you do not have port conflicts between WebSphere Application Server - Express instances and products.
- The prerequisite validator verifies your WebSphere Application Server installation and determines whether or not the prerequisite software is installed.

Application development scripts

You can use these scripts when you develop applications:

- The dumpNameSpace script dumps the contents of the name space accessed through a name server.
- The JspBatchCompiler script compiles JSP files as a batch.

Web services scripts

- The Java2WSDL script uses the Java API for XML-based remote procedure call (JAX-RPC) specification to map a Java class to a Web Services Description Language (WSDL) file.
- The setupWebServiceClientEnv script sets up a Java environment for Web Services J2SE clients to use and sets the classpath variable for Web Services clients.
- The wsdeploy script adds Websphere Application Server - Express product-specific Web services deployment classes to a Web services enterprise archive (EAR) file or an application client Java archive (JAR) file.
- The WSDL2Java script uses the Java API for XML-based remote procedure call (JAX-RPC) specification to create Java classes and deployment descriptor templates from a Web Services Description Language (WSDL) file.

Security scripts

- "Encode passwords in properties files" on page 48 encodes passwords in a properties file.
- "Encode password data" on page 48 generates a text file that contains user IDs and passwords for a client that does not pass a user ID and password on a getConnection call.

Configure Qshell to run WebSphere Application Server - Express scripts

Most of the WebSphere Application Server - Express scripts are located in the /QIBM/ProdData/WebASE/ASE5/bin directory. The crtpluginst script is located in the /QIBM/ProdData/WebASE/bin directory. These scripts must be run from Qshell. There are several ways to run Qshell commands to ensure the correct version (from the correct directory) is used.

- Invoke the script from the CL command line or from an i5/OS CL program. To use this method, run the STRQSH command and specify the fully qualified path name of the script:

```
STRQSH CMD('/QIBM/ProdData/WebASE/ASE5/bin
  /script_name parameters')
```

where *script_name* is the name of the script and *parameters* represents the parameters that are passed to the script.

- From the Qshell prompt, invoke the fully qualified path name of the script:

  ```
  /QIBM/ProdData/WebASE/ASE5/bin/script_name parameters
  ```

  where *script_name* is the name of the script and *parameters* represents the parameters that are passed to the script.
- From the Qshell prompt, use the cd command to change to the /QIBM/ProdData/WebASE/ASE5/bin directory, and then run the script:

  ```
  cd /QIBM/ProdData/WebASE/ASE5/bin

  script_name parameters
  ```
- You can also update the PATH environment variable to automatically locate the script when you run it. After you update the PATH variable, you can run these scripts from any directory. To update the PATH environment variable, follow these steps:
  1. Edit the .profile file in the /home/*user_profile_name* directory, where *user_profile_name* is the name of your iSeries user profile.

     Note: If this file does not exist, create it in this directory. You can use the EDTF command from an CL command line or use any editor from a workstation. Also note that `.profile` is the full name of the file. When you start Qshell, it searches for the .profile file, and runs the commands listed in it. You can use the .profile file to set persistent environment variables for your Qshell session.
  2. Add this line to the .profile file:

     ```
     export PATH=/QIBM/ProdData/WebASE/ASE5/bin:/QIBM/ProdData/WebASE/bin:$PATH
     ```
  3. Save the file.

Qshell environment variables

WebSphere Application Server - Express provides Qshell environment variables that affect the scripts.

To set an environment variable for your Qshell session, run this command from the Qshell command line:

```
export variableName=value
```

To unset an environment variable for your Qshell session, run this command from the Qshell command line:

```
unset variableName
```

Note: When you exit the Qshell session, the environment variable must be set again the next time you issue the STRQSH CL command.

Note: You can also use a profile file to set variables. For more information, see "Configure Qshell to run WebSphere Application Server - Express scripts" on page 92.

- **WAS_ADDL_JVM_ARGS**
  If you set this variable, its value is appended to the Java virtual machine (JVM) arguments for all of the scripts that run a JVM. (All of the scripts except crtwasinst, dltwasinst, updwashost, enbprfwas, startServer, and lstwasinst run a JVM.) For example, this command:

  ```
  export WAS_ADDL_JVM_ARGS="-Dtrace=com.ibm.*=all=enabled -Xms256m"
  ```

  enables tracing for the JVM that a script starts, and sets the minimum heap size for the JVM to 256 megabytes.
- **WAS_USER_SCRIPT**
  This variable specifies the path to a script that runs before a WebSphere Application Server - Express Qshell script runs. For example, if you want to append a /home/QSYS/classes.jar to the classpath for every script, you might create a script called /home/myDir/classpath.script. The script file contains this command:

  ```
  WAS_CLASSPATH=${WAS_CLASSPATH}:/home/QSYS/classes.jar
  ```

After you create the script file, run this command to set the environment variable:

```
export WAS_USER_SCRIPT=/home/myDir/classpath.script
```

Security and Qshell scripts

If security is enabled for your application server, you must provide authentication information to the scripts. If security is enabled and you do not provide authentication information, the scripts receive an AccessDenied exception. You can provide authentication data in any of these ways:

- Most command line tools support -username and -password options that you can use to provide basic authentication data. The userid and password that you specify should be an administrative user. For example, you can use a member of the WebSphere administrative console users with operator or administrator privileges, or the administrative userid configured in the user registry. In this example, the stopServer script specifies the required command line parameters:

  ```
  stopServer -username adminuser -password adminpw
  ```
- You can place the authentication data in a properties file that the command line tools read.
  - If you use the SOAP connector, the default file for this data is the soap.client.props.
  - If you use the RMI connector, use the sas.client.props file.

  The script uses the properties files in the /QIBM/UserData/WebASE/ASE5/*instance*/properties directory.

  where *instance* is the instance name specified by the -instance parameter for the command.

Set explicit authorities for the startServer and stopServer scripts

You can set explicit authorities to allow a user profile to start and stop application servers even if the profile does not have *ALLOBJ authority. When you set these authorities for a user profile, that user profile can also use the HTTP Administration forms to start and stop application servers.

Note: In the steps below, *userid* is the user profile for which you want to set the authorities.

Set authorities for the startServer script

To set explicit authorities for the startServer script, follow these steps:

1. Run these commands on an CL command line:
   a.
      ```
      CHGAUT OBJ('/QIBM/ProdData/WebASE/ASE5/bin/startServer') USER(USERID) DTAAUT(*RX)
      ```
   b.
      ```
      CHGUSRPRF USRPRF(userid) SPCAUT(*JOBCTL)
      ```
   c.
      ```
      GRTOBJAUT OBJ(QASE5/STRSVRWAIT) OBJTYPE(*PGM) USER(USERID) AUT(*USE)
      ```
   d.
      ```
      GRTOBJAUT OBJ(QASE5/QASE5) OBJTYPE(*JOBD) USER(USERID) AUT(*USE)
      ```
   e.
      ```
      GRTOBJAUT OBJ(QASE5/QASE5) OBJTYPE(*JOBQ) USER(USERID) AUT(*USE)
      ```
   f.
      ```
      GRTOBJAUT OBJ(QSYS/QEJBSVR) OBJTYPE(*USRPRF) USER(USERID) AUT(*USE)
      ```
   g. If QEJBSVR is not authorized to the output queue of the *userid* user profile, you must grant QEJBSVR *USE authority to the output queue:
      ```
      GRTOBJAUT OBJ(outqlib/outqname) OBJTYPE(*OUTQ) USER(QEJBSVR) AUT(*USE)
      ```
      If you do not want to grant explicit authority to the user's output queue, create an output queue to which QEJBSVR is authorized and use the SBMJOB CL command to start the server. Specify the appropriate value for the OUTQ parameter on the SBMJOB command.
2. Run the Start Qshell (STRQSH) command from the CL command line.

3. In Qshell, run these commands:
   a.
      ```
      /QIBM/ProdData/WebASE/ASE5/bin/grtwasaut -instance instance -user userid
        -dtaaut rx
      ```
   b.
      ```
      /QIBM/ProdData/WebASE/ASE5/bin/grtwasaut -instance instance -user userid
        -dtaaut rx -object bin -recursive
      ```

After you perform these steps, the user profile can start the server from Qshell with either of these commands:

-
  ```
  /QIBM/ProdData/WebASE/ASE5/bin/startServer -instance instance -nowait
  ```
  This command starts the default server (*instance*) for instance *instance*.

-
  ```
  SBMJOB CMD(CALL PGM(QEJBAS5/QASESTRSVR) PARM('-instance'
    '/QIBM/UserData/WebASE/ASE5/instance' '-server' 'server')) JOB(server)
    JOBD(QASE5/QASE5) JOBQ(QASE5/QASE5) USER(QEJBSVR) LANGID(*USRPRF)
    CNTRYID(*USRPRF) CCSID(*USRPRF)
  ```
  where *instance* is the name of the instance and servername is the name of the server.

  This command starts the default server (*instance*) for instance *instance*.

Note: The givedescriptor API used by The STRSVRWAIT program uses the givedescriptor API. This API requires *ALLOBJ in certain situations. As a result, if your user profile does not have *ALLOBJ authority, you must include the -nowait parameter when you run the startServer script.

Set authorities for the stopServer script

To set explicit authorities for the stopServer script, follow these steps:
1. On an CL command line, run this command:
   ```
   CHGAUT OBJ('/QIBM/ProdData/WebASE/ASE5/bin/stopServer') USER(USERID) DTAAUT(*RX)
   ```
2. Run the Start Qshell (STRQSH) command from the CL command line.
3. In Qshell, run these commands:
   a.
      ```
      /QIBM/ProdData/WebASE/ASE5/bin/grtwasaut -instance instance -user userid
        -dtaaut rx
      ```
   b.
      ```
      /QIBM/ProdData/WebASE/ASE5/bin/grtwasaut -instance instance -user userid
        -dtaaut rx -object bin -recursive
      ```
   c.
      ```
      /QIBM/ProdData/WebASE/ASE5/bin/grtwasaut -instance instance -user userid
        -dtaaut rx -object properties
      ```
   d.
      ```
      /QIBM/ProdData/WebASE/ASE5/bin/grtwasaut -instance instance -user userid
        -dtaaut rx -object properties/java.security
      ```

After you perform these steps, the user profile can run this command from Qshell to stop the server:
```
/QIBM/ProdData/WebASE/ASE5/bin/stopServer -instance instance
```

This command stops the default server (*instance*) for instance *instance*.

The wsadmin administrative tool

WebSphere Application Server - Express provides a command line administrative tool named wsadmin, which you can use to run administrative commands interactively or through the use of Jacl script files. The wsadmin tool uses the Bean Scripting Framework (BSF), which supports a variety of scripting languages to configure and control WebSphere Application Server - Express. In WebSphere Application Server - Express, wsadmin supports only the Jacl scripting language.

The wsadmin launcher makes Java objects available through language specific interfaces. Scripts use these objects for application management, configuration, operational control, and for communication with MBeans running in WebSphere server processes.

WebSphere Application Server - Express System Management separates administrative functions into these categories:

- **Configuration:** These functions are related to the configuration of WebSphere Application Server - Express installations.
- **Operation:** These functions are related to the currently running objects in WebSphere Application Server - Express installations.
- **Application management:** These functions are related to installing, uninstalling and managing enterprise applications.

These topics provide information about wsadmin and how you can use it to manage your instance of WebSphere Application Server - Express:

"Run the wsadmin tool"
This topic describes how to start an interactive wsadmin session and other methods to invoke wsadmin commands. You can use an interactive wsadmin session to enter wsadmin commands to administer your instance of WebSphere Application Server - Express.

"Syntax and parameters" on page 98
To run commands and scripts in wsadmin, you must run the `wsadmin` command from Qshell. See this topic for information on the syntax and parameters of the `wsadmin` command.

"Use wsadmin in a secure environment" on page 100
This topic describes how to run wsadmin when security is enabled for your application server.

"Scripting objects" on page 100
The command line administrative client provides four objects that you can use in scripts: AdminApp, AdminControl, AdminConfig, and Help. This topic provides information about each of these objects.

"Java properties of wsadmin" on page 127
The Java properties that wsadmin uses are stored in the wsadmin.properties file. This topic describes these properties.

"Sample wsadmin commands" on page 129
This topic provides links to examples of how you can use wsadmin.

"The Jacl scripting language" on page 129
Jacl is a Java-based implementation of the Tcl scripting language. The wsadmin tool supports the use of Jacl scripts. This topic describes Jacl and provides links to more information.

Run the wsadmin tool

This topic describes how to start and stop an interactive wsadmin session and other ways to invoke wsadmin commands.

The interactive session is a shell environment in which you can run wsadmin commands and scripts.
- Start wsadmin (page 97)
- Stop wsadmin (page 97)

You can also run a single command or script file without starting an interactive session, or run commands in a profile before the interactive session starts. The other ways that you can invoke wsadmin commands are:
- Run wsadmin commands individually (page 97)
- Run wsadmin commands in a script (page 97)
- Run wsadmin commands in a profile (page 98)

Start wsadmin

To start an interactive wsadmin session, follow these steps:
1. Enter the Start Qshell (STRQSH) command on an CL command line.
2. Use the cd command to change to the bin directory of the product installation root:
   ```
   cd /QIBM/ProdData/WebASE/ASE5/bin
   ```
3. At the Qshell prompt, enter this command:
   ```
   wsadmin -instance instance
   ```
   where *instance* is the name of the instance that you want to administer.

Before you exit the interactive session, run this command to save your configuration changes:
```
$AdminConfig save
```

If you do not run this command, wsadmin discards your changes when you exit the interactive session.

Stop wsadmin

To exit wsadmin, enter this command at the wsadmin prompt:
```
exit
```

Run wsadmin commands individually

To run commands individually, specify the -c option when you run the `wsadmin` command.
```
wsadmin -instance instance -c '$AdminApp list'
WASX7209I: Connected to process "instance" on node node using
SOAP connector;  The type of process is: UnManagedProcess
"Business Applications"
DB2WebServicesSamples
ExpressSamples
"IBM Telephone Directory"
adminconsole
```

In this example, *instance* is the name of your WebSphere Application Server - Express instance, *server* is the name of your application server, and *node* is the name iSeries host server:

When you run commands with the -c parameter, configuration changes are saved automatically. You do not need to run the `$AdminConfig save` command.

Run wsadmin commands in a script

After you run a script, wsadmin returns you to the Qshell prompt.

To run commands in a script, specify the -f option when you run the `wsadmin` command:

```
wsadmin -instance instance -f 'script.jacl'

WASX7209I: Connected to process "server" on node node using SOAP connector;
The type of process is: UnManagedProcess
 adminconsole
 DefaultApplication
 ivtApp
```

In this example, *instance* is the name of your WebSphere Application Server - Express instance, *server* is the name of your application server, *node* is the name of the node where your instance runs, and *script.jacl* is the fully qualified path of a Jacl script that contains these commands:

```
set apps [$AdminApp list]
puts $apps
```

If you run a script file that makes changes to your application server's configuration, you must include the `$AdminConfig save` command at the end of the script. If you do not include this command, wsadmin does not save your changes.

Run wsadmin commands in a profile

The wsadmin tool starts an interactive session after it runs the commands in a profile.

To run commands in a profile, specify the -profile option when you run the `wsadmin` command.

```
wsadmin -instance instance -profile 'profile.jacl'

WASX7209I: Connected to process "server" on node node using SOAP connector;
The type of process is: UnManagedProcess
 Applications currently installed:
 adminconsole
 DefaultApplication
 ivtApp
 WASX7029I: For help, enter: "$Help help"
 wsadmin>
```

In this example, *instance* is the name of your WebSphere Application Server - Express instance, *server* is the name of your application server, *node* is the name of the node where your instance runs, and *profile.jacl* is the fully qualified path of a profile that contains these commands:

```
set apps [$AdminApp list]
puts "Applications currently installed:\n$apps"
```

Syntax and parameters

This page describes the syntax and parameters of the wsadmin command.

Syntax

```
wsadmin -instance instance [ -c 'command' | -f scriptfile ] [ -javaoption option ]
 [ -lang lang ] [ -p propertiesfile ] [ -profile scriptfile ]
 [ -conntype SOAP | RMI | JMS | NONE [ -host host ] [ -port port] ]
 [ -wsadmin_classpath classpath ] [ -help | -? ] [ script_parameters ]
```

Parameters

- **-instance**
  This required parameter specifies the instance that you want to administer with wsadmin. If you specify only this parameter, the script starts an interactive wsadmin session.
- **-c**
  This optional parameter specifies a single command to run. If you specify this parameter, wsadmin runs the command that you enter and then returns control to the Qshell prompt. To run multiple commands, specify a -c parameter for each command that you want to run. The commands run in the order in which they are listed.

- **-f**
  This optional parameter specifies a script to run. If you specify this parameter, wsadmin runs the script that you specify and then returns you to the Qshell prompt.
- **-javaoption**
  This optional parameter is available in versions 5.0.2 and later. The value option specifies a valid Java standard or non-standard option. You can include more than one -javaoption parameter when you run the wsadmin command.
- **-lang**
  Because Jacl is the only supported scripting language, you do not need to specify this parameter.
- **-p**
  This optional parameter specifies a properties file to load. Three properties files are loaded before any files specified with -p.
  1. The first file is the wsadmin.properties file in the /QIBM/UserData/WebASE/ASE5/*instance*/properties directory, where *instance* is the name of your application server instance.
  2. The second file is /home/QEJBSVR/wsadmin.properties. This file is not shipped with WebSphere Application Server - Express.
  3. The third file is the file specified by the WSADMIN_PROPERTIES environment variable. This variable is not set by WebSphere Application Server - Express.

  If you specify the -p parameter, the specified properties file overrides properties in the instance default file and the user default file. You can specify the -p parameter multiple times. If you do, the properties files are invoked in the order specified. For example, if the com.ibm.ws.scripting.port property is specified in the instance default file, the user default, and a custom properties file that you specify with the -p parameter, the script uses the value in the custom properties file.
- **-profile**
  This optional parameter specifies a profile script. The profile script runs before other commands or scripts. If you specify -c, the profile runs before the single command. If you specify -f, the profile runs before the script. If you do not specify -c or -f, wsadmin runs the commands in the profile and then starts an interactive wsadmin session. You can specify multiple -profile parameters. If you do, the wsadmin script invokes the profile scripts in the order that you specify them.
- **-conntype**
  This optional parameter specifies to type of connection to use to connect to the application server that you want to administer. The valid values for the conntype parameter are SOAP, RMI, and NONE. The default value is SOAP. The -conntype parameter determines which port, if any, is used to connect to the server. If you specify NONE as the conntype parameter, only the $AdminApp commands are available.
- **-host**
  This optional parameter specifies the host name of the system running the application server that you want to administer. If you do not specify this parameter, the host name defaults to localhost.
- **-port**
  If you specify the -conntype parameter, you can specify the port number to use to connect to the remote application server. For example, if you use a SOAP connection, specify the SOAP port for the application server you want to administer. If you do not specify this parameter, the script uses the value specified for the `com.ibm.ws.scripting.port` property in the wsadmin.properties file for your instance, if that property exists. If the property does not exist in any properties file specified explicitly or implicitly for the wsadmin script, an error occurs. When an instance is created, the wsadmin.properties file is updated with the SOAP and RMI connector ports for the instance.
- **-wsadmin_classpath**
  This optional parameter makes additional classes available to your scripting process. The value *classpath* specifies the classpath that contains the classes that you want wsadmin to access. The specified classpath is added to the classloader for the scripting process. You can also specify this option in a wsadmin.properties file with the com.ibm.ws.scripting.classpath property. The -wsadmin_classpath parameter overrides any value specified in wsadmin.properties.

- **-help** or **-?**
  Optional argument that provides syntax help.
- **script_parameters**
  If you specify the -f parameter to run a script, you can include that script's parameters in the wsadmin command.

Examples

The following examples demonstrate correct syntax. In these examples, mymachine is the name of the host on which server containing the SOAP or RMI connector is running. The port is specified in the wsadmin.properties file by com.ibm.ws.scripting.port.

- SOAP connection to the default WebSphere Application Server - Express instance on the local host
  ```
  wsadmin
  ```
- SOAP connection to the WebSphere Application Server - Express instance test on the local host
  ```
  wsadmin -instance test
  ```
- SOAP connection with port 8880 to the as400 host
  ```
  wsadmin -conntype SOAP -host as400 -port 8880
  ```
- RMI connection with port 2809 to the as400 host
  ```
  wsadmin -conntype RMI -host as400 -port 2809
  ```

Use wsadmin in a secure environment

If you enable security for a WebSphere Application Server cell, you must supply authentication information when you invoke wsadmin. The wsadmin tool allows you to provide authentication information in any of these ways:

- Update the soap.client.props file for Simple Object Access Protocol (SOAP) (page 100)
- Specify authentication information when you run a command (page 100)

Update the soap.client.props file for Simple Object Access Protocol (SOAP)

If you use a Simple Object Access Protocol (SOAP) connector, set these properties in the soap.client.props file:

```
com.ibm.SOAP.loginUserid=user
com.ibm.SOAP.loginPassword=password
```

where *user* is you user profile and *password* is your password.

Specify authentication information when you invoke wsadmin

You can specify the -user and -password parameters when you invoke wsadmin. For example, to use the SOAP connector, run this command from the Qshell prompt:

```
wsadmin -conntype SOAP -port SOAPport -user user -password password
```

where *SOAPport* is the port number that your application server uses for SOAP, *user* is you user profile, and *password* is your password.

If you specify user and password information on a command line and in the soap.client.props file, the command line information overrides the information in the properties file.

Scripting objects

These topics provide information about each of the objects that you can use in scripts for wsadmin.

"The AdminControl object for scripted administration" on page 101
You can use the AdminControl object to invoke operational commands on objects that run in WebSphere Application Server - Express. Many of the AdminControl methods have multiple

signatures so that you can invoke them in a raw mode using parameters specified by Java Management Extensions (JMX), or using strings for parameters.

"The AdminConfig object for scripted administration" on page 110
Use the AdminConfig object to invoke configuration commands and create or change elements of WebSphere Application Server - Express configuration.

"The AdminApp object for scripted administration" on page 122
The AdminApp object interacts with WebSphere Application Server - Express management and configuration services to install and manage applications.

"The Help object for scripted administration" on page 126
The Help object provides general help and dynamic online information about the MBeans that are running. You can also use the Help object as an aid in writing and running scripts with the AdminControl object. Some methods include: attributes, operations, AdminConfig, and AdminControl.

The AdminControl object for scripted administration: Use the AdminControl object to invoke operational commands on objects that run in WebSphere Application Server. Many of the AdminControl methods have multiple signatures. You can invoke these methods in a raw mode with parameters specified by Java Management Extensions (JMX), or with strings for parameters. It is recommended that you use string signatures. In addition to operational commands, the AdminControl object supports some utility methods for tracing, reconnecting with a server, and converting data types.

For additional information about the AdminControl object, see Example: Collecting arguments for the AdminControl object.

completeObjectName
Creates a string representation of a complete ObjectName value based based on a fragment. This method does not communicate with the server to find a matching ObjectName value. If it finds several MBeans that match the fragment, the method returns the first matching MBean.

Parameters: name

Example usage:

```
set serverON [$AdminControl completeObjectName node=mynode,type=Server,*]
```

getAttribute
Returns the value of the attribute for the name you provide.

Parameters: name, attribute

Example usage:

```
set objNameString [$AdminControl completeObjectName WebSphere:type=Server,*]
$AdminControl getAttribute $objNameString processType
```

getAttribute_jmx
Returns the value of the attribute for the name you provide.

Parameters: name, attribute

Example usage:

```
set objNameString [$AdminControl completeObjectName WebSphere:type=Server,*]
set objName [java::new javax.management.ObjectName $objNameString]
$AdminControl getAttribute_jmx $objNameString processType
```

getAttributes
Returns the attribute values for the names you provide.

Parameters: name, attributes

Example usage:

```
set objNameString [$AdminControl completeObjectName WebSphere:type=Server,*]
$AdminControl getAttributes $objName "cellName nodeName"
```

getAttributes_jmx
Returns the attribute values for the names you provide.

Parameters: name, attributes

Example usage:

```
set objectNameString [$AdminControl completeObjectName WebSphere:type=Server,*]
set objName [java::;new javax.management.ObjectName $objectNamestring]
set attrs [java::new {String[]} 2 {cellName nodeName}]
$AdminControl getAttributes_jmx $objName $attrs
```

getCell
Returns the name of the connected cell.

Parameters: none

Example usage:

```
$AdminControl getCell
```

Example output:

```
Mycell
```

getConfigId
Creates a configuration ID from an ObjectName or ObjectName fragment. Use this ID with the $AdminConfig method. Not all Mbeans that run have configuration objects that correspond. If there are several Mbeans that correspond to an ObjectName fragment, the command returns a warning message and creates a configuration ID for the first Mbean that it finds.

Parameters: name

Example usage:

```
set threadpoolCID [$AdminControl getConfigId node=mynode,type=ThreadPool,*]
```

getDefaultDomain
Returns the default domain name from the server.

Parameters: none

Example usage:

```
$AdminControl getDefaultDomain
```

Example output:

```
WebSphere
```

getDomainName
Returns the domain name from the server.

Parameters: none

Example usage:
```
$AdminControl getDomainName
```

Example output:
```
WebSphere
```

getHost
Returns the name of your host.

Parameters: none

Example usage:
```
$AdminControl getHost
```

Example output:
```
myhost
```

getMBeanCount
Returns the number of Mbeans registered in the server.

Parameters: none

Example usage:
```
$AdminControl getMBeanCount
```

Example output:
```
114
```

getMBeanInfo_jmx
Returns the JMX MBeanInfo structure that corresponds to an OjbectName value. There is no string signature for this method, because the Help object displays most of the information available from getMBeanInfo.

Parameters: name

Example usage:
```
set objName [java::new javax.management.ObjectName [$AdminControl
  completeObjectName type=Server,*]]
$AdminControl getMBeanInfo_jmx $objName
```

Example output:
```
javax.management.modelmbean.ModelMBeanInfoSupport@10dd5f35
```

getNode
Returns the name of the connected node.

Parameters: none

Example usage:
```
$AdminControl getNode
```

Example output:

```
myhost
```

getPort
Returns the name of your port.

Parameters: none

Example usage:

```
$AdminControl getPort
```

Example output:

```
8877
```

getPropertiesForDataSource
This command is deprecated in version 5.0.2. The command incorrectly assumes the availability of a configuration service when running in connected mode.

getType
Returns the connection type.

Parameters: none

Example usage:

```
$AdminControl getType
```

Example output:

```
SOAP
```

help
Returns general help text for the AdminControl object.

Parameters: none

Usage:

```
$AdminControl help
```

Output:

```
WASX7027I: The AdminControl object enables the manipulation
of MBeans running in a WebSphere server process.  The number and type
of MBeans available to the scripting client depends on the server to
which the client is connected.  If the client is connected to a
Deployment Manager, then all the MBeans running in the Deployment
Manager are visible, as are all the MBeans running in the Node Agents
connected to this Deployment Manager, and all the MBeans running in
the application servers on those nodes.

The following commands are supported by AdminControl; more detailed
information about each of these commands is available by using the
"help" command of AdminControl and supplying the name of the command
as an argument.

Note that many of these commands support two different sets of
signatures: one that accepts and returns strings, and one low-level
set that works with JMX objects like ObjectName and AttributeList.
In most situations, the string signatures are likely to be more useful,
but JMX-object signature versions are supplied as well.  Each of these
JMX-object signature commands has "_jmx" appended to the command name.
```

```
Hence there is an "invoke" command, as well as a "invoke_jmx" command.

completeObjectName
                Return a String version of an object name given a
                template name
getAttribute_jmx
                Given ObjectName and name of attribute, returns value of
                attribute
getAttribute    Given String version of ObjectName and name of attribute,
                returns value of attribute
getAttributes_jmx
                Given ObjectName and array of attribute names, returns
                AttributeList
getAttributes   Given String version of ObjectName and attribute names,
                returns String of name value pairs

getCell         returns the cell name of the connected server
getConfigId     Given String version of ObjectName, return a config id for
                the corresponding configuration object, if any.
getDefaultDomain
                returns "WebSphere"
getDomainName   returns "WebSphere"

getHost         returns String representation of connected host
getMBeanCount   returns number of registered beans
getMBeanInfo_jmx
                Given ObjectName, returns MBeanInfo structure for MBean

getNode         returns the node name of the connected server
getPort         returns String representation of port in use
getType         returns String representation of connection type in use
help            Show help information
invoke_jmx      Given ObjectName, name of method, array of parameters and
                signature, invoke method on MBean specified
invoke          Invoke a method on the specified MBean
isRegistered_jmx
                true if supplied ObjectName is registered
isRegistered    true if supplied String version of ObjectName is registered
makeObjectName  Return an ObjectName built with the given string
queryNames_jmx  Given ObjectName and QueryExp, retrieves set of ObjectNames
                that match.
queryNames      Given String version of ObjectName, retrieves String of
                ObjectNames that match.
reconnect       reconnects with server
setAttribute_jmx
                Given ObjectName and Attribute object, set attribute for MBean
                specified
setAttribute    Given String version of ObjectName, attribute name and
                attribute value, set attribute for MBean specified
setAttributes_jmx
                Given ObjectName and AttributeList object, set attributes for
                the MBean specified
startServer     Given the name of a server, start that server.
stopServer      Given the name of a server, stop that server.
testConnection  Test the connection to a DataSource object
trace           Set the wsadmin trace specification
```

help

Returns help text for the specific method of the AdminControl object. The method name is not case sensitive.

Parameters: method

Example usage:

```
$AdminControl help getAttribute
```

Example output:

```
WASX7043I: Method: getAttribute; Arguments: object name, attribute Description:
Returns value of "attribute" for the MBean described by "object name."
```

invoke
Invokes the object operation without parameters. Returns the result of the invocation.

Parameters: name, operationName

Example usage:

```
set objNameString [$AdminControl completeObjectName WebSphere:type=Server,*]
$AdminControl invoke $objNameString stop
```

invoke
Invokes the object operation based on the parameter list that you supply. The signature generates automatically. The types of parameters are supplied by examining the MBeanInfo that the MBean supplies. Returns the string result of the invocation.

Parameters: name, operationName

Example usage:

```
set objNameString [$AdminControl completeObjectName WebSphere:type=Server,*]
$AdminControl invoke $objNameString appendTraceString com.ibm.*=all=enabled
```

invoke
Invokes the object operation by conforming the parameter list to the signature. Returns the result of the invocation.

Parameters: name, operationName, params, sigs

Example usage:

```
set objNameString [$AdminControl completeObjectName WebSphere:type=Server,*]
$AdminControl invoke $objNameString appendTraceString com.ibm.*=all=enabled
  java.lang.String
```

invoke_jmx
Invokes the object operation by conforming the parameter list to the signature. Returns the result of the invocation.

Parameters: name, operationName, params, sigs

Example usage:

```
set objNameString [$AdminControl completeObjectName WebSphere:type=TraceService,*]
set objName [java::new javax.management.ObjectName $objNameString]
set parms [java::new {java.lang.Object[]} 1 com.ibm.ejs.sm.*=all=disabled]
set signature [java::new {java.lang.String[]} 1 java.lang.String]
$AdminControl invoke_jmx $objName appendTraceString $parms $signature
```

isAlive

Parameters: none

Example usage:

```
$AdminControl isAlive
```

isInstanceof
If the ObjectName value is a member of the class you provide, then the value is true.

Parameters: name, class name

Example usage:

```
set objNameString [$AdminControl completeObjectName WebSphere:type=Server,*]
$AdminControl isInstanceOf $objNameString java.lang.Object
```

isInstanceof_jmx
If the ObjectName value is a member of the class you provide, then the value is true.

Parameters: name, class name

Example usage:

```
set objName [java::new javax.management.ObjectName WebSphere:type=Server,*]
$AdminControl isInstanceOf_jmx $objName java.lang.Object
```

isRegistered
If the ObjectName value is registered in the server, then the value is true.

Parameters: name

Example usage:

```
set objNameString [$AdminControl completeObjectName WebSphere:type=Server,*]
$AdminControl isRegistered $objNameString
```

isRegistered_jmx
If the ObjectName value is registered in the server, then the value is true.

Parameters: name

Example usage:

```
set objName [java::new javax.management.ObjectName WebSphere:type=Server,*]
$AdminControl isRegistered_jmx $objName
```

makeObjectName
Creates an ObjectName value based on the strings that you specify. This method does not communicate with the server, and may generate an ObjectName value that does not exist. If the string you supply contains an extra set of quotation marks (“), they are removed. If the string does not begin with a Java Management eXtensions (JMX) domain, or a string followed by a colon, then the WebSphere string prepends to the name.

Parameters: name

Example usage:

```
set objName [$AdminControl makeObjectName WebSphere:type=Server,node=mynode,*]
```

queryNames
Returns a string that lists all ObjectNames based on the name template.

Parameters: name

Example usage:

```
$AdminControl queryNames WebSphere:type=Server,*
```

Example output:

```
WebSphere:cell=BaseApplicationServerCell,name=server1,mbeanIdentifier=server1,
  type=Server,node=mynode,process=server1
```

queryNames_jmx
Returns a set of ObjectName objects, based on the ObjectName and QueryExp that you provide.

Parameters: name, query

Example usage:

```
set objName [java::new javax.management.ObjectName WebSphere:type=Server,*]
set null [java::null]
$AdminControl queryNames_jmx $objName $null
```

Example output:

```
[WebSphere:cell=BaseApplicationServerCell,name=server1,mbeanIdentifier=server1,
  type=Server,node=mynode,process=server1]
```

reconnect
Reconnects to the server, and clears information out of the local cache.

Parameters: none

Example usage:

```
$AdminControl reconnect
```

Example output:

```
WASX7074I: Reconnect of SOAP connector to host myhost completed.
```

setAttribute
Sets the attribute value for the name you provide.

Parameters: name, attributeName, attributeValue

Example usage:

```
set objNameString [$AdminControl completeObjectName
  WebSphere:type=TraceService,*]
$AdminControl setAttribute $objNameString traceSpecification
  com.ibm.*=all=disabled
```

setAttribute_jmx
Sets the attribute value for the name you provide.

Parameters: name, attribute

Example usage:

```
set objectNameString [$AdminControl completeObjectName
  WebSphere:type=TraceService,*]
set objName [java:;new javax.management.ObjectName $objectNamestring]
set attr [java::new javax.management.Attribute
  traceSpecification com.ibm.*=all=disabled]
$AdminControl setAttribute_jmx $objName $attr
```

setAttributes
Sets the attribute values for the names you provide and returns a list of successfully set names.

Parameters: name, attributes

Example usage:

```
set objNameString [$AdminControl completeObjectName
  WebSphere:type=TracesService,*]
$AdminControl setAttributes $objNameString {{traceSpecification
  com.ibm.ws.*=all=enabled}}
```

setAttributes_jmx
Sets the attribute values for the names you provide and returns a list of successfully set names.

Parameters: name, attributes

Example usage:

```
set objectNameString [$AdminControl completeObjectName
  WebSphere:type=TraceService,*]
set objName [java:;new javax.management.ObjectName $objectNamestring]
set attr [java::new javax.management.Attribute traceSpecification
  com.ibm.ws.*=all=enabled]
set alist [java::new javax.management.AttributeList]
$alist add $attr
$AdminControl setAttributes_jmx $objName $alist
```

startServer
Starts the specified application server. This command uses the default wait time. You can only use this command if the scripting client is connected to a NodeAgent. It returns a message to indicate if the server starts successfully.

Parameters: server name

Example usage:

```
$AdminControl startServer server1
```

startServer
Starts the specified application server. The start process waits the number of seconds specified by the wait time for the server to start. You can only use this command if the scripting client is connected to a NodeAgent. It returns a message to indicate if the server starts successfully.

Parameters: server name, wait time

Example usage:

```
$AdminControl startServer server1 100
```

stopServer
Stops the specified application server. The command returns a message to indicate if the server stops successfully.

Parameters: server name

Example usage:

```
$AdminControl stopServer server1
```

stopServer
Stops the specified application server. If you set the flag to immediate, the server stops immediately. The command returns a message to indicate if the server stops successfully.

Parameters: server name, immediate flag

Example usage:

```
$AdminControl stopServer server1 immediate
```

stopServer
Stops the specified application server. The command returns a message to indicate if the server stops successfully.

Parameters: server name, node name

Example usage:

```
$AdminControl stopServer server1 myNode
```

stopServer
Stops the specified application server. If you set the flag to immediate, the server stops immediately. The command returns a message to indicate if the server stops successfully.

Parameters: server name, node name, immediate flag

Example usage:

```
$AdminControl stopServer server1 myNode immediate
```

testConnection
Communicates with the DataSourceCfgHelper Mbean to test a DataSource connection. This command works with DataSource resided in the configuration repository. If the DataSource to be tested is in the temporary workspace that holds the update to the repository, you must save the update to the configuration repository before you run this command. Use this method with the configuration ID that corresponds to the DataSource and the WAS40DataSource object types. The return value is a message that indicates a successful connection or a connection with warning. If the connection fails, an exception is thrown from the server.

Parameters: configId

Note: In versions 5.0.2 and later, the **props** parameter is not supported for this command.

Example usage:

```
set ds [lindex [$AdminConfig list DataSource] 0]
$AdminControl testConnection $ds
```

Example output:

```
WASX7217I: Connection to provided datasource was successful.
```

trace
Sets the trace specification for the scripting process to the value that you specify.

Parameters: traceSpec

Example usage:

```
$AdminControl trace com.ibm.ws.scripting.*=all=enabled
```

The AdminConfig object for scripted administration: Use the AdminConfig object to invoke configuration commands and to create or change elements of the WebSphere Application Server configuration.

You can start the scripting client without a running server, if you only want to use local operations. To run in local mode, use the -conntype NONE option to start the scripting client. You will receive a message that you are running in the local mode. If an application server is running, it is recommended that you do not run the AdminConfig tool in local mode.

The following public methods are available for the AdminConfig object:

attributes
Returns a list of the top level attributes for a given type.

Note: The name of the object type that you input here is based on the XML configuration files and does not need to be the same name that the administrative console displays.

Parameters: object type

Example usage:

```
$AdminConfig attributes ApplicationServer
```

Example output:

```
"properties Property*" "serverSecurity ServerSecurity" "server Server@" "id Long"
"stateManagement StateManageable" "name String" "moduleVisibility
EEnumLiteral(MODULE, COMPATIBILITY, SERVER, APPLICATION)"
"services Service*" "statisticsProvider StatisticsProvider"
```

checkin
Checks a file in to the configuration repository. The file is described by the document URI.

Note: This method only applies to deployment manager configurations.

Parameters: document URI, filename, opaque object

Example usage:

```
$AdminConfig checkin cells/MyCell/Node/MyNode/serverindex.xml
  /home/mydir/myfile $obj
```

The document URI is relative to the root of the configuration repository (/QIBM/UserData/WebASE/ASE5/*instance*/config, where *instance* is the name of your application server instance). The file specified by the filename parameter is used as the source of the file to check in. The opaque object is an object that the extract command of the AdminConfig object returns by a prior call.

contents
Obtains information about object types.

Note: The name of the object type that you input here is the one based on the XML configuration files and does not have to be the same name that the administrative console displays.

Parameters: object type

Example usage:

```
$AdminConfig contents JDBCProvider
```

Example output:

```
{DataSource DataSource}
{WAS40DataSource WAS40DataSource}
```

convertToCluster
Converts a server so that it is the first member of a new ServerCluster.

Arguments: server id, cluster name

Example usage:

```
set serverid [$AdminConfig getid /Server:myServer/]
   $AdminConfig convertToCluster $serverid myCluster
```

Example output:

```
myCluster(cells/mycell/clusters/myCluster:cluster.xml#ClusterMember_2
```

create
Creates configuration objects.

Note: The name of the object type that you input here is the one based on the XML configuration files. It does not have to be the same name that the administrative console displays.

Parameters: type, parent ID, attributes

Example usage:

```
set jdbc1 [$AdminConfig getid /JDBCProvider:jdbc1/]
$AdminConfig create DataSource $jdbc1 {{name ds1}}
```

Example output:

```
ds1(cells/mycell/nodes/DefaultNode/servers/server1:resources.xml#DataSource_6)
```

createClusterMember
Creates a new server as a member of an existing cluster.

This method creates a new server object on the node that the node id argument specifies. This server is created as a new member of the existing cluster specified by the cluster id argument, and contains attributes specified in the member attributes argument. The server is created using the server template specified by the template id attribute, and contains the name specified by the memberName attribute. The memberName attribute is required.

Arguments: cluster id, node id, member attributes

Note: The name of the object type that you specify here is the one based on the XML configuration files. It does not need to be the same name that the administrative console displays.

Example usage:

```
set clid [$AdminConfig getid /ServerCluster:myCluster/]
set nodeid [$AdminConfig getid /Node:mynode/]
set template [$AdminConfig getid /Node:mynode/Server:myServer/]
$AdminConfig createClusterMember $clid $nodeid {{memberName newMem1} {weight 5
$template
```

Example output:

```
myCluster(cells/mycell/clusters/myCluster:cluster.xml#ClusterMember_2)
```

createDocument
Creates a new document in the configuration repository.

The documentURI argument names the document to create in the repository. The filename argument must be a valid local file name where the contents of the document exist.

Parameters: documentURI, filename

Example usage:

```
$AdminConfig createDocument cells/mycell/myfile.xml /home/mydir/myfile
```

createUsingTemplate
Creates a type of object with the given parent, using a template.

Parameters: type, parent id, attributes, template ID

Example usage:

```
set node [$AdminConfig getid /Node:mynode/]
set templ [$AdminConfig listTemplates JDBCProvider "DB2 JDBC Provider (XA)"]
$AdminConfig createUsingTemplate JDBCProvider $node {{name newdriver}} $templ
```

defaults
Displays the default values for attributes of a given type.

This method displays all of the possible attributes contained by an object of a specific type. If the attribute has a default value, this method also displays the type and default value for each attribute.

Note: The name of the object type that you input here is the one based on the XML configuration files. It does not have to be the same name that the administrative console displays.

Parameters: type

Example usage:

```
$AdminConfig defaults TuningParams
```

Example output:

```
 Attribute                      Type                           Default
 usingMultiRowSchema            Boolean                        false
 maxInMemorySessionCount        Integer                        1000
 allowOverflow                  Boolean                        true
 scheduleInvalidation           Boolean                        false
 writeFrequency                 ENUM
 writeInterval                  Integer                        120
 writeContents                  ENUM
 invalidationTimeout            Integer                        30
 invalidationSchedule           InvalidationSchedule
```

deleteDocument
Deletes a document from the configuration repository.

The documentURI argument names the document that will be deleted from the repository.

Parameters: documentURI

Example usage:

```
$AdminConfig deleteDocument cells/mycell/myfile.xml
```

existsDocument
Tests for the existence of a document in the configuration repository.

The documentURI argument names the document to test in the repository.

Parameters: documentURI

Example usage:

```
$AdminConfig existsDocument cells/mycell/myfile.xml
```

Example output:

```
1
```

extract
Extracts a configuration repository file described by document URI and places it in the file named by filename.

Note: This method only applies to deployment manager configurations.

Parameters: document URI, filename

Example usage:

```
set obj [$AdminConfig extract cells/MyCell/Node/MyNode/serverindex.xml
  /home/mydir/myfile]
```

The document URI is relative to the root of the configuration repository (/QIBM/UserData/WebASE/ASE5/*instance*/config, where *instance* is the name of your application server instance). If the file specified by filename exists, the extracted file replaces it.

getCrossDocumentValidationEnabled
Returns a message with the current cross-document enablement setting.

This method returns true if cross-document validation is enabled.

Parameters: none

Example usage:

```
$AdminConfig getCrossDocumentValidationEnabled
```

Example output:

```
WASX7188I: Cross-document validation enablement set to true
```

getid
Returns the configuration id of an object.

Parameters: containment path

Example usage:

```
$AdminConfig getid /Cell:testcell/Node:testNode/JDBCProvider:Db2JdbcDriver/
```

Example output:

```
Db2JdbcDriver(cells/testcell/nodes/testnode/resources.xml#JDBCProvider_1)
```

getObjectName
Returns a string version of the object name for the corresponding running MBean.

This method returns an empty string if there is no corresponding running MBean.

Parameters: configuration id

Example usage:

```
set server [$AdminConfig getid /Node:mynode/Server:server1/]
$AdminConfig getObjectName $server
```

Example output:

```
WebSphere:cell=mycell,
name=server1,mbeanIdentifier=cells/mycell/nodes/mynode/servers/
server1/server.xml#Server_1,type=Server,node=mynode,process=server1,
processType=UnManagedProcess
```

getSaveMode
Returns the mode used when you invoke a save command.

Possible values include the following:
- overwriteOnConflict - saves changes even if they conflict with other configuration changes
- rollbackOnConflict - causes a save operation to fail if changes conflict with other configuration changes. This value is the default.

Parameters: none

Example usage:

```
$AdminConfig getSaveMode
```

Example output:

```
rollbackOnConflict
```

getValidationLevel
Returns the validation used when files are extracted from the repository.

Parameters: none

Example usage:

```
$AdminConfig getValidationLevel
```

Example output:

```
WASX7189I: Validation level set to HIGH
```

getValidationSeverityResult
Returns the number of validation messages with the given severity from the most recent validation.

Parameters: severity

Example usage:

```
$AdminConfig getValidationSeverityResult 1
```

Example output:

```
16
```

hasChanges
Returns true if unsaved configuration changes exist.

Parameters: none

Example usage:

```
$AdminConfig hasChanges
```

Example output:

```
1
```

help

Displays static help information for the AdminConfig object.

Parameters: none

Usage:

```
$AdminConfig help
```

Output:

```
WASX7053I: The AdminConfig object communicates with the
Config Service in a WebSphere server to manipulate configuration data
for a WebSphere installation.  AdminConfig has commands to list, create,
remove, display, and modify configuration data, as well as commands to
display information about configuration data types.

Most of the commands supported by AdminConfig operate in two modes:
the default mode is one in which AdminConfig communicates with the
WebSphere server to accomplish its tasks.  A local mode is also
possible, in which no server communication takes place.  The local
mode of operation is invoked by bringing up the scripting client with
no server connected using the command line "-conntype NONE" option
or setting the "com.ibm.ws.scripting.connectionType=NONE" property in
the wsadmin.properties.

The following commands are supported by AdminConfig; more detailed
information about each of these commands is available by using the
"help" command of AdminConfig and supplying the name of the command
as an argument.

attributes      Show the attributes for a given type
checkin         Check a file into the the config repository.
convertToCluster
                converts a server to be the first member of a
                new ServerCluster
create          Creates a configuration object, given a type, a parent, and
                a list of attributes, and optionally an attribute name for the
                new object
createClusterMember
                Creates a new server that is a member of an
                existing cluster.
createDocument  Creates a new document in the config repository.
installResourceAdapter
                Installs a J2C resource adapter with the given rar
                file name and an option string in the node.
createUsingTemplate
                Creates an object using a particular template type.
defaults        Displays the default values for attributes of a given type.
deleteDocument  Deletes a document from the config repository.
existsDocument  Tests for the existence of a document in the config repository.
extract         Extract a file from the config repository.
getCrossDocumentValidationEnabled
                Returns true if cross-document validation is enabled.
getid           Show the configId of an object, given a string version of
                its containment
getObjectName   Given a config id, return a string version of the ObjectName
                for the corresponding running MBean, if any.
getSaveMode     Returns the mode used when "save" is invoked
getValidationLevel
                Returns the validation used when files are extracted from the
                repository.
getValidationSeverityResult
                Returns the number of messages of a given
                severity from the most recent validation.
hasChanges      Returns true if unsaved configuration changes exist
help            Show help information
```

```
list            Lists all configuration objects of a given type
listTemplates   Lists all available configuration templates of a given
                type.
modify          Change specified attributes of a given configuration object
parents         Show the objects which contain a given type
queryChanges    Returns a list of unsaved files
remove          Removes the specified configuration object
required        Displays the required attributes of a given type.
reset           Discard unsaved configuration changes
save            Commit unsaved changes to the configuration repository
setCrossDocumentValidationEnabled
                Sets the cross-document validation enabled mode.
setSaveMode     Changes the mode used when "save" is invoked
setValidationLevel
                Sets the validation used when files are extracted from the
                repository.
show            Show the attributes of a given configuration object
showall         Recursively show the attributes of a given configuration
                object, and all the objects contained within each attribute.
showAttribute   Displays only the value for the single attribute specified.
types           Show the possible types for configuration
validate        Invokes validation
```

installResourceAdapter
Installs a J2C resource adapter with the given RAR file name and an option string in the node.

The RAR file name is the fully qualified file name that resides in the node that you specify. The valid options include the following:

- rar.name
- rar.desc
- rar.archivePath
- rar.classpath
- rar.nativePath

All options are optional. The rar.name option is the name for the J2CResourceAdapter. If you do not specify this option, the display name in the rar deployment descriptor is used. If that is not specified, the RAR file name is used. The rar.desc option is a description of the J2CResourceAdapter. The rar.archivePath is the name of the path where the file is to be extracted. If you do not specify this option, the archive will be extracted to the $\{CONNECTOR_INSTALL_ROOT\} directory. The rar.classpath is the additional class path.

Parameters: rar file name, node, options

Example usage:

```
$AdminConfig installResourceAdapter /rar/mine.rar {-rar.name myResourceAdapter
  -rar.desc "My rar file"} mynode
```

Example output:

```
myResourceAdapter(cells/mycell/nodes/mynode:resources.xml#J2CResourceAdapter_1)
```

list
Returns a list of objects of a given type, possibly scoped by a parent.

Note: The name of the object type that you input here is the one based on the XML configuration files and does not have to be the same name that the administrative console displays.

Parameters: object type

Example usage:

```
$AdminConfig list JDBCProvider
```

Example output:

```
Db2JdbcDriver(cells/mycell/nodes/DefaultNode/resources.xml#JDBCProvider_1)
Db2JdbcDriver(cells/mycell/nodes/DefaultNode/servers/deploymentmgr/
  resources.xml#JDBCProvider_1)
Db2JdbcDriver(cells/mycell/nodes/DefaultNode/servers/nodeAgent/
  resources.xml#JDBCProvider_1)
```

listTemplates
Displays a list of template object IDs.

Parameters: object type

Example usage:

```
$AdminConfig listTemplates JDBCProvider
```

This example displays a list of all JDBCProvider templates available on the system.

modify
Supports modification of object attributes.

Parameters: object, attributes

Example usage:

```
$AdminConfig modify
ConnFactory1(cells/mycell/nodes/DefaultNode/servers/deploymentmgr/resources.xml
#GenericJMSConnectionFactory_1) {{userID newID} {password newPW}}
```

parents
Obtains information about object types.

Note: The name of the object type that you input here is the one based on the XML configuration files and does not have to be the same name that the administrative console displays.

Parameters: object type

Example usage:

```
$AdminConfig parents JDBCProvider
```

Example output:

```
Cell
Node
Server
```

queryChanges
Returns a list of unsaved configuration files.

Parameters: none

Example usage:

```
$AdminConfig queryChanges
```

Example output:

```
WASX7146I: The following configuration files contain unsaved changes:
cells/mycell/nodes/mynode/servers/server1/resources.xml
```

remove
Removes a configuration object.

Parameters: object

Example usage:

```
$AdminConfig remove ds1(cells/mycell/nodes/DefaultNode/servers/server1:
  resources.xml#DataSource_6)
```

required
Displays the required attributes contained by an object of a certain type.

Note: The name of the object type that you input here is the one based on the XML configuration files. It does not have to be the same name that the administrative console displays.

Parameters: object

Example usage:

```
$AdminConfig required URLProvider
```

Example output:

```
Attribute                       Type
streamHandlerClassName          String
protocol                        String
```

reset
Resets the temporary workspace that holds updates to the configuration.

Parameters: none

Example usage:

```
$AdminConfig reset
```

save
Saves changes in the configuration repository.

Parameters: none

Example usage:

```
$AdminConfig save
```

setCrossDocumentValidationEnabled
Sets the cross-document validation enabled mode. Values include true or false.

Parameters: flag

Example usage:

```
$AdminConfig setCrossDocumentValidationEnabled true
```

setSaveMode
Allows you to toggle the behavior of the save command. The default is rollbackOnConflict. When a conflict is discovered while saving, the unsaved changes are not committed. The alternative is overwriteOnConflict which saves the changes to the configuration repository even if there are conflicts.

Parameters: mode

Example usage:

```
$AdminConfig setSaveMode overwriteOnConflict
```

setValidationLevel
Sets the validation used when files are extracted from the repository.

There are five validation levels: none, low, medium, high, or highest.

Parameters: level

Example usage:

```
$AdminConfig setValidationLevel high
```

Example output:

```
WASX7189I: Validation level set to HIGH
```

show
Returns the top level attributes of the given object.

Parameters: object, attributes

Example usage:

```
$AdminConfig show Db2JdbcDriver(cells/mycell/nodes/DefaultNode/
  resources.xm#JDBCProvider_1)
```

Example output:

```
{name "Sample Datasource"} {description "Data source for the Sample entity beans"}
```

showall
Recursively shows the attributes of a given configuration object.

Parameters: object, attributes

Example usage:

```
$AdminConfig showall "Default
Datasource(cells/mycell/nodes/DefaultNode/servers/server1:
  resources.xml#DataSource_1)
```

Example output:

```
{authMechanismPreference BASIC_PASSWORD}
{category default}
{connectionPool {{agedTimeout 0}
{connectionTimeout 1000}
{maxConnections 30}
{minConnections 1}
{purgePolicy FailingConnectionOnly}
{reapTime 180}
{unusedTimeout 1800}}}
{datasourceHelperClassname com.ibm.websphere.rsadapter.CloudscapeDataStoreHelper}
{description "Datasource for the WebSphere Default Application"}
{jndiName DefaultDatasource}
{name "Default Datasource"}
{propertySet {{resourceProperties {{{description "Location of Cloudscape
  default database."}
{name databaseName}
{type java.lang.String}
{value ${WAS_INSTALL_ROOT}/bin/DefaultDB}} {{name remoteDataSourceProtocol}
{type java.lang.String}
{value {}}} {{name shutdownDatabase}
{type java.lang.String}
{value {}}} {{name dataSourceName}
{type java.lang.String}
{value {}}} {{name description}
{type java.lang.String}
{value {}}} {{name connectionAttributes}
{type java.lang.String}
{value {}}} {{name createDatabase}
{type java.lang.String}
{value {}}}}}}}
{provider "Cloudscape JDBC
Driver(cells/pongo/nodes/pongo/servers/server1:resources.xml#JDBCProvider_1)"}
{relationalResourceAdapter "WebSphere Relational Resource
Adapter(cells/pongo/nodes/pongo/servers/server1:resources.xml#builtin_rra)"}
{statementCacheSize 0}
```

showAttribute
Displays only the value for the single attribute that you specify.

The output of this command is different from the output of the show command when a single attribute is specified. The showAttribute command does not display a list that contains the attribute name and value. It only displays the attribute value.

Parameters: config id, attribute

Example usage:

```
set ns [$AdminConfig getid /Node:mynode/]
$AdminConfig showAttribute $n hostName
```

Example output:

```
mynode
```

types
Returns a list of the configuration object types that you can manipulate.

Parameters: none

Example usage:

```
$AdminConfig types
```

Example output:

```
AdminService
Agent
ApplicationConfig
ApplicationDeployment
ApplicationServer
AuthMechanism
AuthenticationTarget
AuthorizationConfig
AuthorizationProvider
AuthorizationTableImpl
BackupCluster
CMPConnectionFactory
CORBAObjectNameSpaceBinding
Cell
CellManager
```

```
Classloader
ClusterMember
ClusteredTarget
CommonSecureInteropComponent
```

validate
Invokes validation.

This command requests configuration validation results based on the files in your workspace, the value of the cross-document validation enabled flag, and the validation level setting. The scope of this request is the object named by the config id argument.

Parameters: config id

Example usage:

```
$AdminConfig validate
```

Example output:

```
WASX7193I: Validation results are logged in
  /QIBM/UserData/WebASE/ASE5/myInstance/logs/wsadmin.valout:
Total number of messages: 16
WASX7194I: Number of messages of severity 1: 16
```

The AdminApp object for scripted administration: The AdminApp object interacts with the WebSphere Application Server management and configuration services to administer applications. For example, you can use the AdminApp object to install and uninstall applications, list the modules for an application, and export an application.

You can invoke most of the AdminApp functions in local mode, which means that the client does not communicate with the server to accomplish the function. To run in local mode, use the -conntype NONE option when you start the scripting client. If a server is running, it is not recommended that you run the AdminApp tool in local mode.

For additional information about the AdminApp object, see these topics:
- Installation options for the AdminApp object
- Example: Obtaining information about task options for the AdminApp install command

The following public methods are available for the AdminApp object:

deleteUserAndGroupEntries
Deletes users or groups for all roles, and deletes userids and passwords for all of the RunAs roles defined in the application.

Parameters: appname

Example usage:

```
$AdminApp deleteUserAndGroupEntries myapp
```

edit
Edits an application or module in interactive mode.

Parameters: appname, options

Example usage:

```
$AdminApp edit "JavaMail Sample" {-MapWebModToVH {{"JavaMail
Sample WebApp" mtcomps.war,WEB-INF/web.xml newVH}}}
```

Note: The edit command changes the application deployment. Specify these changes in the options parameter. No options are required for the edit command.

editInteractive
Edits an application or module in interactive mode.

Parameters: appname, options

Example usage:

```
$AdminApp editInteractive ivtApp
```

Note: The editInteractive command changes the application deployment. Specify these changes in the options parameter. No options are required for the editInteractive command.

export
Exports the application appname parameter to a file you specify by file name.

Parameters: appname, filename

Example usage:

```
$AdminApp export "My App" /usr/me/myapp.ear
```

exportDDL
Extracts the description definition language (DDL) from the application appname parameter to the directoryname parameter that a directory specifies.

Note: This command is not supported when running from a system with only standalone scripting client install.

Parameters: appname, directoryname, options

Example usage:

```
$AdminApp exportDDL "My App" /usr/me/DDL
```

help
Displays help for the AdminApp object and AdminApp commands.

Output:

```
wsadmin>$AdminApp help
WASX7095I: The AdminApp object allows application objects to
be manipulated including installing, uninstalling, editing,
and listing.  Most of the commands supported by AdminApp operate in two
modes: the default mode is one in which AdminApp communicates with the
WebSphere server to accomplish its tasks.  A local mode is also
possible, in which no server communication takes place.  The local
mode of operation is invoked by including the "-conntype NONE" flag in the
option string supplied to the command.

The following commands are supported by AdminApp; more detailed
information about each of these commands is available by using the
"help" command of AdminApp and supplying the name of the command
```

```
as an argument.

edit            Edit the properties of an application
editInteractive Edit the properties of an application interactively
export          Export application to a file
exportDDL       Extract DDL from application to a directory
help            Show help information
install         Installs an application, given a file name and an option string.
installInteractive
                Installs an application in interactive mode, given a file name
                and an option string.
list            List all installed applications
listModules     List the modules in a specified application
options         Shows the options available, either for a given file, or in
                general.
taskInfo        Shows detailed information pertaining to a given install task
                for a given file
uninstall       Uninstalls an application, given an application name and
                an option string
```

install
Installs an application in non-interactive mode, given a fully qualified file name and a string of installation options.

Note: This command is not supported when running from a system with only standalone scripting client install.

Parameters: earfile, options

Example usage:

```
$AdminApp install /home/mydir/apps/myapp.ear
```

There are many options available for this command. You can obtain a list of valid options for an EAR file with the following command:

```
$AdminApp options <earfilename>
```

You can also obtain help for each object with the following command:

```
$AdminApp help <optionname>
```

installInteractive
Installs an application in interactive mode, given a fully qualified file name and a string of installation options.

Note: This command is not supported when running from a system with only standalone scripting client install.

Parameters: earfile, options

Example usage:

```
$AdminApp installInteractive /QIBM/UserData/WebASE/ASE5/myInstance/installableApps/jmsample.ear
```

list
Lists the applications installed in the configuration.

Parameters: none

Example usage:

```
$AdminApp list
```

Example output:

```
wsadmin>$AdminApp list
adminconsole
DefaultApplication
ivtApp
```

listModules
Lists the modules in an application.

Parameters: appname, options

Example usage:

```
$AdminApp listModules ivtApp
```

Example output:

```
wsadmin>$AdminApp listModules ivtApp
ivtApp#ivtEJB.jar+META-INF/ejb-jar.xml
ivtApp#ivt_app.war+WEB-INF/web.xml
```

This example is formed by the concatenation of appname, #, module URI, +, and DD URI. You can pass this string to the edit and editInteractive AdminApp commands.

options
Displays a list of options for installing an EAR file.

Parameters: earfile

Example usage:

```
$AdminApp options /QIBM/UserData/WebASE/ASE5/myInstance/installableApps/jmsample.ear
```

publishWSDL
Publishes WSDL files for the application specified in the appname parameter to the file specified in the filename parameter using the soap address prefixes specified in the soapAddressPrefixes parameter.

Parameters: appname, filename, soapAddressPrefixes

Example usage:

```
$AdminApp publishWSDL JAXRPCHandlersServer /temp/a.zip
{{JAXRPCHandlersServerApp.war  {{http http://localhost:9080}}}}
```

taskInfo
Provides information about a particular task option for an application file.

Parameters: earfile, task name

Example usage:

```
$AdminApp taskInfo /QIBM/UserData/WebASE/ASE5/myInstance/installableApps/jmsample.ear
  MapWebModToVH
```

Example output:

```
MapWebModToVH: Selecting virtual hosts for Web modules
Specify the virtual host where you want to install the Web modules contained in
your application. Web modules can be installed on the same virtual host
or dispersed among several hosts.
Each element of the MapWebModToVH task consists of the following 3 fields:
"webModule," "uri," "virtualHost."
Of these fields, the following may be assigned new values: "virtualHost"
```

```
and the following are required: "virtualHost"

The current contents of the task after running default bindings are:
webModule: JavaMail Sample WebApp
uri: mtcomps.war,WEB-INF/web.xml
virtualHost: null
```

updateAccessIDs
Updates the access id information for users and groups assigned to various roles defined in the application. The access ids are read from the user registry and saved in the application bindings. It is recommended that you call this method after you install an application or after you edit security role information for an installed application. The scripting client must be connected to the application server when you invoke this command.

The bAll boolean parameter retrieves and saves all access IDs for users and groups in the application bindings. Specify false if you want to retrieve access ids for users or groups that do not have an access id in the application bindings.

Parameters: appname, bAll

Example usage:

```
$AdminApp updateAccessIDs myapp true
```

The Help object for scripted administration: The Help object provides general help and dynamic online information about the MBeans that are running in your application server instance. You can use the Help object when your write and run scripts with the AdminControl object.

The following public methods are available for the Help object:

AdminApp
Provides a summary of all of the available methods for the AdminApp object.

AdminConfig
Provides a summary of all of the available methods for the AdminConfig object.

AdminControl
Provides a summary of all of the available methods for the AdminControl object.

all
Provides a summary of the information that the MBean defines by name.

attributes
Provides a summary of all of the attributes that the MBean defines by name.

classname
Provides a class name that the MBean defines by name.

constructors
Provides a summary of all of the constructors that the MBean defines by name.

description
Provides a description that the MBean defines by name.

help
Provides a summary of all of the available methods for the help object.

message
Displays information for a message ID.

Parameters: message ID

Example usage:
```
$Help message WASX7015E
```

Example output:
```
 Explanation: An error occurred during the interactive execution of the
  specified command.
User action: Examine the accompanying exception information to determine the
  appropriate action.
```

notifications
Provides a summary of all the notifications that the MBean defines by name.

operations
Provides a summary of all of the operations that the MBean defines by name.

operations
Provides the signature of the opname operation for the MBean defined by name.

Java properties of wsadmin

The wsadmin scripting client uses the Java properties listed below. These properties are specified in the wsadmin.properties file, which is located in the /QIBM/UserData/WebASE/ASE5/*instance*/properties directory, where *instance* is the name of your instance.

You can also specify a properties file for wsadmin when you run the wsadmin script. There are three levels of default properies files that load before any properties file specified on the command line. These properties files load in the order that is shown below. Each properties file overrides settings from previously loaded files.

1. The first level, located in the WebSphere Application Server - Express instance's properties directory is called wsadmin.properties. This file is located in the /QIBM/UserData/WebASE/ASE5/*instance* properties directory, where *instance* is the name of your application server instance.
2. The second level is in /home/QEJBSVR/wsadmin.properties. This file and the directories that contain it are not shipped with WebSphere Application Server - Express. You must create the directories and files, and assign appropriate authorities.
3. The third level is a file specified by the WSADMIN_PROPERTIES environment variable. This environment variable is defined in the environment where the wsadmin tool starts. Generally, this is the QShell session in which you run the wsadmin script. WebSphere Application Server - Express does not define this environment variable.
4. Lastly, properties specified on the command line are processed. This mean that command line properties override values specified in any of the properties files.

Note: Some of these properties have two default values.

- **iSeries default value**
  This value is set in the wsadmin.properties file when you create an instance. Values specified in this file override the wsadmin tool default.
- **wsadmin tool default value**
  These values are hardcoded in the tool. If no other values are specified, the tool uses these default values.

These are the Java properties contained in the wsadmin.properties file:

- **com.ibm.ws.scripting.connectionType**
  This value determines the type of connection to use. Possible values are SOAP (Simple Object Access Protocal), RMI (Remote Method Invocation), or NONE. The iSeries default is SOAP.
- **com.ibm.ws.scripting.port**
  This property specifies the port that the connector uses. The default value is the SOAP port for your instance.
- **com.ibm.ws.scipting.host**
  The host to which the scripting process attempts to connect. The default is localhost.
- **com.ibm.ws.scripting.defaultLang**
  This property specifies the language to use when executing scripts. Jacl is the only supported language. If you specify the -f parameter, wsadmin checks to see if the file extension maps to a know value. For example, if the file name is test.jacl it will set the language to JACL. If the -f parameter is not specified or the file type is unknown, wsadmin generates an error.
  - iSeries default value: jacl.
  - Tool default value: none
- **com.ibm.ws.scripting.traceFile**
  This property specifies the file to which the scripting process writes trace and logging information.
  - iSeries default value: the wsadmin.traceout file in the /QIBM/UserData/WebASE/ASE5/*instance*/logs directory.
  - Tool default value: Print trace output to the display.
- **com.ibm.ws.scripting.validationOutput**
  This property specifies where validation reports are directed. If multiple users simultaneously administer an instance with wsadmin, it is recommended that you specify different validationOutput properties in user properties files.
  - iSeries default value: The wsadmin.valout file in the /QIBM/UserData/WebASE/ASE5/*instance*/logs directory.
  - Tool default value: wsadmin.valout in the current directory.
- **com.ibm.ws.scripting.traceString**
  This property is used to turn on tracing for the scripting process. For example, use com.ibm.ws.scripting.*=all=enabled to turn on all tracing for the the scripting code. The default is to not use tracing.
- **com.ibm.ws.scripting.profiles**
  The profiles property is a list of profiles that wsadmin runs before it runs user commands, scripts, or an interactive shell.
  - The iSeries default value includes /QIBM/ProdData/WebASE/ASE5/bin/securityProcs.jacl and /QIBM/ProdData/WebASE/ASE5/bin/LTPA_LDAPSecurityProcs.jacl. These profile scripts make it easier to configure security.
  - Tool default value: none
- **com.ibm.ws.scripting.emitWarningForCustomSecurityPolicy**
  This property specifes whether message WASX7207W is emitted when custom permissions are found. The possible values are TRUE and FALSE.
  - iSeries default value: none
  - Tool default value: TRUE
- **com.ibm.ws.scripting.tempdir**
  This property specifies the directory to use for temporary files when installing applications.
  - iSeries default value: none
  - Tool default value: the value of Java system property java.io.tempdir
- **com.ibm.ws.scripting.validationLevel**
  This property specifies the level of validation to use when configuration changes are made from the scripting interface. Possible values are NONE, LOW, MEDIUM, HIGH, HIGHEST.

  - iSeries default value: none
  - Tool default value: HIGHEST
- **com.ibm.ws.scripting.crossDocumentValidationEnabled**
  This property specifies whether the validation mechanism examines other documents when changes are made to one document. Possible values are TRUE and FALSE.
  - iSeries default value: none
  - Tool default value: TRUE
- **com.ibm.ws.scripting.classpath**
  The classpath property is appended to the list of paths to search for classes and resources. There is no default value for iSeries or the wsadmin tool.

Sample wsadmin commands

These help topics provide examples of how you can use wsadmin to configure and manage application servers, applications, and resources in WebSphere Application Server.

- Configuration management examples with wsadmin
- Application management examples with wsadmin
- Operation management examples with wsadmin
- Scripting management examples with wsadmin

The Jacl scripting language

Jacl is an alternate implementation of TCL, and is written entirely in Java code.

The basic syntax for a Jacl command is:

```
Command   arg1   arg2   arg3   ...
```

The command is either the name of a built-in command or a Jacl procedure. For example:

```
puts stdout  {Hello, world!}
```

This example demonstrates the `puts` command, which takes two arguments: an I/O stream identifier (stdout) and a string (Hello, world!). The `puts` command writes the string and a trailing new line character to the I/O stream. In Jacl, the command interprets the specified arguments. The use of stdout as a name is a convention employed by puts and the other I/O commands. Use stderr to identify the standard error output, and use stdin to identify the standard input.

For more information on Tcl and Jacl, see these articles:

- Tcl for WebSphere Application Server - Express administrators
- Jacl: A Tcl implementation in Java

The ws_ant script

Apache Ant is a Java-based build tool. WebSphere Application Server - Express provides a set of Ant-based tasks that you can use to perform some common administrative operations. A Qshell script called ws_ant is provided to run these tasks.

Note: This function is a technology preview.

For more information on Ant, see the Apache Jakarta Ant page.

These topics provide information on using ws_ant:

"Start ws_ant"
To run WebSphere Application Server - Express Ant tasks, run the ws_ant command from Qshell. See this topic for information on how to invoke the ws_ant command.

"Syntax and parameters"
See this topic for information on the syntax and parameters of the ws_ant command.

"Ant tasks" on page 131
See this topic for information on the Ant tasks provided.

Start ws_ant

To run a WebSphere Application Server - Express ANT task using the ws_ant script, follow these steps:

1. Enter the Start Qshell (STRQSH) command on an CL command line.
2. Use the cd command to change to the /QIBM/ProdData/WebASE/ASE5/bin directory.
3. At the Qshell prompt, enter this command:

   ```
   ws_ant target -buildfile buildfile -instance instance
   ```

   where *target* is the name of the target to run, *buildfile* is name of the file that contains the target definition, and *instance* is the name of the instance you want to administer.

See "Syntax and parameters" for information on other parameters available with the ws_ant script.

See "Ant tasks" on page 131 for information on the tasks provided.

Syntax and parameters

For more information about running ANT commands, see the Apache ANT User Manual.

Authority

To run this script your user profile must have *ALLOBJ authority.

Syntax

The syntax of the ws_ant script is:

```
ws_ant [ target ] [ -instance instance ] [-find file]
 [-buildfile buildfile] [-listener listenerclass]
 [-logger loggerclass] [-logfile logfile] [-emacs] [-debug]
 [-verbose] [-quiet] [-version] [-projecthelp] [-help]
```

Parameters

The parameters of the ws_ant script are:

- ***target***
  This is an optional parameter. The value *target* specifies the name of the target to execute in the buildfile. If this parameter is not specified, the script calls the default target of your buildfile.
- **-instance**
  This is an optional parameter. The value *instance* specifies the name of your instance. The default value is `default`.
- **-find**
  This is an optional parameter. The value *file* specifies to search towards the root of the file system for the buildfile to be used. For example, if you run the ws_ant command from /QIBM/ProdData/WebASE/ASE5/bin and specify -find myfile.xml, the script searches the file system in this order:
  1. /QIBM/ProdData/WebASE/ASE5/bin/myfile.xml
  2. /QIBM/ProdData/WebASE/ASE5/myfile.xml
  3. /QIBM/ProdData/WebASE/myfile.xml
  4. /QIBM/ProdData/myfile.xml
  5. /QIBM/myfile.xml
  6. /myfile.xml
- **-buildfile**
  This is an optional parameter. The value *buildfile* specifies the fully qualified name of the buildfile that contains the target definition. The default value is build.xml.
- **-listener**
  This is an optional parameter. The value *listenerclass* specifies a class name to be added as a project listener.
- **-logger**
  This is an optional parameter. The value *loggerclass* specifies a class name to be used for logging.
- **-logfile**
  This is an optional parameter. The value *logfile* specifies the name of the file that to which the script writes logging information.
- **-emacs**
  This is an optional parameter. By default, ANT adds the name of the current task to the beginning of all logging messages. If you specify the -emacs parameter, the script does not add the name of the task to the messages.
- **-debug**
  This is an optional parameter. If you specify this parameter, the script displays debugging information.
- **-verbose**
  This optional parameter turns on verbose messages, which can be helpful if you need to debug the script.
- **-quiet**
  This is an optional parameter. If you specify this parameter, the script does not display any progress messages.
- **-version**
  This is an optional parameter. If you specify -version, the script displays version information.
- **-projecthelp**
  This optional parameter displays the project help message.
- **-help**
  This optional parameter displays the syntax help message.

Ant tasks

The ws_ant tool supports these tasks:

- **wsadmin**
  The wsadmin task runs the command-line administration tool with the specified arguments. The implementation class for this task is com.ibm.websphere.ant.tasks.WsAdmin.

  You can run this task from a remote machine that has the WebSphere Application Server - Express administrative tools installed.

  Note: When you use the wsadmin scripting objects, you must include an additional dollar sign ($) before the name of the object. For example, if you want ws_ant to invoke the $AdminApp list command to list the applications on your application server, include this task in the ws_ant buildfile:

  ```
  <taskdef name="wsAdmin" classname="com.ibm.websphere.ant.tasks.WsAdmin"/>
    <target name="target">
      <wsAdmin command="$$AdminApp list"/>
    </target>
  ```

  where *target* is a value that is passed on the ws_ant call.
- **wsInstallApp**
  The wsInstallApp task installs an application into an application server or a cell. The implementation class for this task is com.ibm.websphere.ant.tasks.InstallApplication.

  Because wsadmin cannot perform remote installations, you must run this task on the local machine.
- **wsJspC**
  The wsJspC task compiles a directory of JSP files into .class files. The implementation class for this task is com.ibm.websphere.ant.tasks.JspC.

  You must run this task on the local machine.
- **wsListApps**
  The wsListApps task lists all the applications installed on an application server or a cell. The implementation class for this task is com.ibm.websphere.ant.tasks.ListApplications.

  You can run this task from a remote machine that has the WebSphere Application Server - Express administrative tools installed.
- **wsNLSEcho**
  The wsNLSEcho task provides message retrieval with translation and variable substitution. The implementation class for this task is com.ibm.websphere.ant.tasks.NLSEcho.

  You must run this task on the local machine.
- **wsServerStatus**
  The wsServerStatus task displays the status of one application server or all application servers. The implementation class for this task is com.ibm.websphere.ant.tasks.ServerStatus.

  You must run this task on the local machine.
- **wsStartApp**
  The wsStartApp task starts an application in an application server or cell. The implementation class for this task is com.ibm.websphere.ant.tasks.StartApplication.

You can run this task from a remote machine that has the WebSphere Application Server - Express administrative tools installed.

- **wsStartServer**
  The wsStartServer task starts an application server. The implementation class for this task is com.ibm.websphere.ant.tasks.StartServer.

  You must run this task on the local machine.
- **wsStopApp**
  The wsStopApp task stops an application in an application server or cell. The implementation class for this task is com.ibm.websphere.ant.tasks.StopApplication.

  You can run this task from a remote machine that has the WebSphere Application Server - Express administrative tools installed.
- **wsStopServer**
  The wsStopServer task stops an application server. The implementation class for this task is com.ibm.websphere.ant.tasks.StopServer.

  You must run this task on the local machine.
- **wsUninstallApp**
  The wsUninstallApp task uninstalls an application from an application server or cell. The implementation class for this task is com.ibm.websphere.ant.tasks.UninstallApplication.

  Because wsadmin cannot uninstall applications remotely, you must run this task on the local machine.
- **wsValidateModule**
  The wsValidateModule task validates the deployment descriptor, extensions, and bindings documents of an EAR, WAR, EJB Jar, or Application Client Jar. The implementation class for this task is com.ibm.websphere.ant.tasks.ModuleValidator.

  You must run this task on the local machine.

Java Management Extensions

This section describes how to use Java Management Extensions (JMX), which are a set of Java administrative APIs used to customize WebSphere Application Server - Express administration. WebSphere Application Server - Express supports access to the administrative functions through a set of Java classes and methods. You can write a Java program that performs any of the administrative features of the WebSphere Application Server - Express administrative tools. You can also extend basic WebSphere Application Server - Express administration to include your own managed resources. For more information, see the following resources:

- JMX Javadoc
- Java Management Extensions (JMX)

See these topics for information on using JMX with WebSphere Application Server - Express:

“Customize using JMX”
This topic describes how to manage your WebSphere Application Server - Express environment and deploy applications with Java administrative APIs.

“Customize using JMX MBeans” on page 136
This topic describes how to extend WebSphere Application Server - Express administration with JMX MBeans.

“Example: J2EE security permissions” on page 138
This example gives you the parameters necessary to grant J2EE security permissions in order to call JMX methods.

Customize using JMX

Your application or operating environment may have management features that are not accessible in the provided WebSphere Application Server - Express administration facilities. You can incorporate WebSphere Application Server - Express administration into other Java programs by using the Java administrative APIs. Alternatively, you can use the administrative classes and methods to add newly managed objects to WebSphere Application Server - Express administration.

Perform the following steps to develop a Java program using the WebSphere Application Server - Express administrative APIs to access WebSphere Application Server - Express administration:

1. **Develop an administrative client program**.
   a. Create an AdminClient instance.

      An administrative client program needs to invoke methods on the AdminService object. The AdminClient resides in the application server. The AdminClass class contains interfaces defined in the JMX javax.management.MBeanServer interface. See Interface MBeanServer

      for more information.

      The following example shows how to create an AdminClient instance that uses the SOAP connector to interface with the AdminService that runs in the application server:

```
Properties connectProps = new Properties();
connectProps.setProperty
 (AdminClient.CONNECTOR_TYPE, AdminClient.CONNECTOR_TYPE_SOAP);

connectProps.setProperty(AdminClient.CONNECTOR_HOST, "localhost");
connectProps.setProperty(AdminClient.CONNECTOR_PORT, "8880");
AdminClient adminClient = null;
try
{
       adminClient = AdminClientFactory.createAdminClient(connectProps);
}
catch (ConnectorException e)
{
       System.out.println("Exception creating admin client: " + e);
}
```

   b. Find an MBean.

      After you obtain an AdminClient instance, you can use it to access managed resources in an application server. Each managed resource registers an MBean with the AdminService through which you can access the resource. The MBean is represented by an ObjectName instance that identifies the MBean. An ObjectName consists of a domain name followed by an unordered set of one or more key properties. For WebSphere Application Server - Express, the domain name is WebSphere and the key properties defined for administration are as follows:

      - type: The type of MBean. For example: Server, TraceService, JVM. See Public MBean Interfaces for a list of available MBeans.

- name: The name identifier for the individual instance of the MBean.
- cell: The name of the cell in which the MBean is running.
- node: The name of the node in which the MBean is running.
- process: The name of the process in which the MBean is running.

You can locate MBeans by querying for them with ObjectNames that match desired key properties. The following example shows how to find the MBean for the NodeAgent of node MyNode:

```
String nodeName = "MyNode";
String query = "WebSphere:type=NodeAgent,node=" + nodeName + ",*";
ObjectName queryName = new ObjectName(query);
ObjectName nodeAgent = null;
Set s = adminClient.queryNames(queryName, null);
if (!s.isEmpty())
    nodeAgent = (ObjectName)s.iterator().next();
else
    System.out.println("Node agent MBean was not found");
```

c. Use the MBean.

What a particular MBean allows you to do depends on that MBean's management interface. It may declare attributes that you can obtain or set. It may declare operations that you can invoke. It may declare notifications for which you can register listeners. For a list of available MBeans, see Public MBean Interfaces

.

The following example invokes one of the operations available on the NodeAgent MBean that was located above. The following example starts the MyServer application server:

```
String opName = "launchProcess";
String signature[] = { "java.lang.String" };
String params[] = { "MyServer" };
try
{
     adminClient.invoke(nodeAgent, opName, params, signature);
}
catch (Exception e)
{
     System.out.println("Exception invoking launchProcess: " + e);
}
```

d. Register for events.

In addition to managing resources, the JMX APIs also support application monitoring for specific administrative events. Refer to the JMX javadoc

for more information on the JMX APIsl. For example, certain events produce notifications when a server starts. Administrative applications can register as listeners for these notifications. WebSphere Application Server - Express provides a full implementation of the JMX notification model and provides additional function so you can receive notifications in a distributed environment.

The following is an example of how an object can register itself for event notifications emitted from an MBean using the node agent ObjectName:

```
adminClient.addNotificationListener(nodeAgent, this, null, null);
```

In this example, the null value results in receiving all of the node agent MBean event notifications. You can also use the null value with the handback object.

e. Handle the events.

Objects receive JMX event notifications via the handleNotification method, which is defined by the NotificationListener interface, and any event receiver must implement. For details on the JMX javax.management.NotificationListener interface, see Interface NotificationListener

.

The following example is an implementation of handleNotification that reports the notifications that it receives:

```
public void handleNotification(Notification n, Object handback)
{
System.out.println("***************************************************");
System.out.println("* Notification received at " + new Date().toString());
System.out.println("* type      = " + ntfyObj.getType());
System.out.println("* message   = " + ntfyObj.getMessage());
System.out.println("* source    = " + ntfyObj.getSource());
System.out.println("* seqNum    =
            " + Long.toString(ntfyObj.getSequenceNumber()));
System.out.println("* timeStamp = " + new Date(ntfyObj.getTimeStamp()));
System.out.println("* userData  = " + ntfyObj.getUserData());
System.out.println("***************************************************");
}
```

2. **Build the administrative client program** by compiling it with javac and providing the location of the necessary JAR files in the class path argument.

   This is an example of a typical command:

```
javac -Djava.version=1.3 -classpath
/QIBM/ProdData/WebASE/ASE5/lib/admin.jar;/QIBM/ProdData/WebASE/ASE5/lib/wsexception.jar;
 /QIBM/ProdData/WebASE/ASE5/lib/jmxc.jar MyAdminClient.java
```

3. **Run the administrative client program** by setting up the runtime environment so that the program can find all of the necessary requirements.

   You can use the /QIBM/ProdData/WebASE/ASE5/bin/setupCmdLine script to set up the environment. This example batch file invokes the setupCmdLine script and runs an administrative client program named MyAdminClient:

```
. $(/usr/bin/dirname $0)/setupCmdLine
TRACE=com.ibm.*=all=disabled TRACEFILE=/home/wasclient/logs/client.log
java ${JAVA_PARM_USER} \
${CONSOLE_ENCODING} \
-Dtrace=$(TRACE) \
-DtraceFile=$(TRACEFILE) \
com.ibm.ws.bootstrap.WSLauncher \
MyAdminClient "$@"
```

Customize using JMX MBeans

You can extend WebSphere Application Server - Express administration by supplying and registering new JMX MBeans in one of the WebSphere Application Server - Express processes. JMX MBeans represent the management interface for a particular piece of logic. All of the managed resources within the standard WebSphere Application Server - Express infrastructure are represented as JMX MBeans. There are a variety of ways to create your own MBeans and register them with the JMX MBeanServer running in any WebSphere process.

Perform the following steps to customize using JMX MBeans:

1. **Create custom JMX MBeans**.

   You can use any existing JMX MBean from another application. You can register any MBean that you tested in a JMX MBeanServer outside of the WebSphere Application Server - Express environment in a WebSphere Application Server - Express process, including Standard MBeans, Dynamic MBeans, Open MBeans, and Model MBeans.

   In addition to any existing JMX MBeans, and ones that were written and tested outside of the WebSphere Application Server - Express environment, you can use the special distributed extensions provided by WebSphere Application Server - Express to create a WebSphere ExtensionMBean provider. This alternative provides better integration with all of the distributed functions of WebSphere Application Server - Express administration. An ExtensionMBean provider implies that you supply an XML file that contains an MBean Descriptor based on the DTD shipped with WebSphere Application Server - Express. The DTD tells WebSphere Application Server - Express all of the attributes, operations, and notifications that your MBean supports. With this information, WebSphere Application Server - Express can route remote requests to your MBean and register remote Listeners to receive your MBean event notifications.

All of the internal WebSphere Application Server - Express MBeans follow the Model MBean pattern. For more information, see Public MBean Interfaces

. Pure Java classes supply the real logic for management functions, and the WebSphere MBeanFactory class reads the description of these functions from the XML MBean Descriptor and creates an instance of a ModelMBean that matches the descriptor. This ModelMBean instance is bound to your Java classes and registered with the MBeanServer running in the same process as your classes. Your Java code becomes callable from any WebSphere Application Server - Express administrative client through the ModelMBean created and registered to represent it.

2. **Register the new MBeans**.

   There are several ways to register your MBean with the MBeanServer in a WebSphere Application Server - Express process. The following list describes the available options:

   - Use the com.ibm.websphere.management.AdminService interface. For more information on this interface, see Interface AdminService

     . You can call the registerMBean() method on the AdminService interface and the invocation is delegated to the underlying MBeanServer for the process, after appropriate security checks. You can obtain a reference to the AdminService using the getAdminService() method of the com.ibm.websphere.management.AdminServiceFactory class.
   - Get MBeanServer instances directly. You can get a direct reference to the JMX MBeanServer instance running in any WebSphere Application Server - Express process, by calling the getMBeanServer() method of the com.ibm.websphere.management.MBeanFactory class. You get a reference to the MBeanFactory class by calling the getMBeanFactory() method of the com.ibm.websphere.management.AdminService interface. Registering the MBean directly with the MBeanServer instance can result in that MBean not participating fully in the distributed features of WebSphere Application Server - Express administration.
   - Go through the com.ibm.websphere.management.MBeanFactory class. If you want the greatest possible integration with WebSphere Application Server - Express, use the MBeanFactory class to manage the life cycle of your MBean through the activateMBean and deactivateMBean methods of the MBeanFactory class. Use these methods by supplying a subclass of the RuntimeCollaborator abstract superclass and an XML MBean descriptor file. Using this approach, you supply a pure Java class that implements the management interface defined in the MBean descriptor. The MBeanFactory class creates the actual ModelMBean and registers it with WebSphere Application Server - Express administration on your behalf.
   - Use the com.ibm.websphere.management.JMXManageable and a CustomService interface. You can make the process of integrating with WebSphere Application Server - Express administration even easier, by implementing a CustomService interface, that also implements the JMXManageable interface. Using this approach, you can avoid supplying the RuntimeCollaborator. When your CustomService interface is initialized, the WebSphere Application Server - Express MBeanFactory class reads your XML MBean descriptor file and creates, binds, and registers an MBean to your CustomService interface automatically. After the shutdown method of your CustomService is called, WebSphere Application Server - Express automatically deactivates your MBean.

Results

Regardless of the approach used to create and register your MBean, you must set up proper J2EE security permissions for your new MBean code. WebSphere Application Server - Express AdminService and MBeanServer are tightly protected using J2EE security permissions, and if you do not explicitly grant your code base permissions, security exceptions are thrown when you attempt to invoke methods of these classes. If you are supplying your MBean as part of your application, you can set the permissions in the was.policy file that you supply as part of your application metadata. If you are using a CustomService interface or other code that is not delivered as an application, you can edit the library.policy file in the node configuration, or even the server.policy file in the properties directory for a specific installation. See "Example: J2EE security permissions" for more information.

Example: J2EE security permissions

You must grant J2EE security permissions to application scoped code for JMX and WebSphere Application Server - Express administrative privileges in order to allow the code to call WebSphere Application Server - Express administrative and JMX methods.

- To invoke JMX class and interface methods, at least one of the following permissions are required:

```
permission com.tivoli.jmx.MBeanServerPermission "MBeanServer.*"
permission com.tivoli.jmx.MBeanServerPermission "MBeanServerFactory.*"
where the individual target names are:
MBeanServer.addNotificationListener
MBeanServer.createMBean
MBeanServer.deserialize
MBeanServer.getAttribute
MBeanServer.getDefaultDomain
MBeanServer.getMBeanCount
MBeanServer.getMBeanInfo
MBeanServer.getObjectInstance
MBeanServer.instantiate
MBeanServer.invoke
MBeanServer.isRegistered
MBeanServer.queryMBeans
MBeanServer.queryNames
MBeanServer.registerMBean
MBeanServer.removeNotificationListener
MBeanServer.setAttribute
MBeanServer.unregisterMBean
MBeanServerFactory.createMBeanServer
MBeanServerFactory.newMBeanServer
MBeanServerFactory.findMBeanServer
MBeanServerFactory.releaseMBeanServer
```

- For WebSphere Application Server - Express administrative APIs, the permissions are the following:

```
permission com.ibm.websphere.management.AdminPermission "getAdminService"
permission com.ibm.websphere.management.AdminPermission "getMBeanFactory"
```

Product library, directories, and subsystems

This page describes the product library, directories, and subsystems that WebSphere Application Server - Express uses on your iSeries server.

Product library and directories

WebSphere Application Server - Express uses the library and directories listed here:

- QIWE and QASE5 libraries
  Product libraries.
- /QIBM/ProdData/WebASE/ASE5 directory
  Product root directory.
- /QIBM/UserData/WebASE/ASE5 directory
  Instance root directory.

Subsystem

Jobs for WebSphere Application Server - Express for iSeries run in the QASE5 subsystem.

Administrative repository

WebSphere Application Server - Express stores configuration data for each application server instance in XML documents, which reside in a cascading hierarchy of directories beneath the root directory for the instance. This hierarchy of directories makes up the administrative repository for a WebSphere Application Server - Express instance. The configuration documents describe the servers, nodes, applications, and resources that are part of the instance.

Hierarchy of configuration directories

The administrative repository for each instance starts with the config directory located directly under the root directory for the instance. For the myAppSvr instance, the administrative respository is contained in the /QIBM/UserData/WebASE/ASE5/myAppSvr/config directory.

The administrative repository for an instance contains the following directories and files:

- **/config**
  The config directory is the repository root directory for an instance. It contains a single file, plugin-cfg-service.xmi. The plugin-cfg-service file defines the custom service that causes the Web server plugin file, plugin-cfg.xml, to be regenerated each time the application server is started.
- **/config/cells**
  The cells subdirectory contains a single subdirectory for the cell to which the instance belongs. The `cells` directory also contains the Web server plugin file, plugin-cfg.xml. The Web server plugin that runs in your HTTP server instance uses this file to determine which web resources are installed in your instance.
- **/config/cells/*cellname***
  For an instance of WebSphere Application Server - Express, the cell name is *hostname_instance*, where *hostname* is the iSeries system host name, and *instance* is the name of the instance.

  The cell directory contains these files, which provide configuration data for the cell and for all of the nodes in the cell:
  - admin-authz.xml
    Contains configuration data for authorizations to administrative functions.
  - cell.xml
    Contains configuration data for the cell.
  - filter.policy
    Enterprise applications use app.policy and was.policy files. The filter.policy file contains permissions that the WebSphere Application Server - Express runtime code removes from the app.policy and was.policy files.
  - integral-jms-authorizations.xml
    This file is not used by WebSphere Application Server - Express.
  - multibroker.xml
    This file is not used by WebSphere Application Server - Express.
  - namestore.xml
    Contains persistent name binding data for the naming service.
  - naming-authz.xml
    Contains the configuration information for authorizations to naming service functions.
  - pmirm.xml
    This file is not used by WebSphere Application Server - Express.

  - resources.xml
    Defines the resources that enterprise applications use. Resources include JDBC providers, data sources, and mail providers.
  - security.xml
    Contains configuration data for the security service.
  - variables.xml
    Contains configuration variables used to specify directory paths. The variables can then be substitiuted for the actual path when specifying locations for log files, JDBC implmentation classes, and application install paths.
  - virtualhosts.xml
    Contains configuration data for virtual hosts and their MIME types.
- **/config/cells/*cellname*/applications**
  The applications subdirectory contains a subdirectory for each application deployed in the cell.
- **/config/cells/*cellname*/applications/*application***
  The names of the applications subdirectories match the names of the application's EAR files. For example, if an application is packaged in App1.ear, its subdirectory is also named App1.ear. Each deployed application subdirectory contains the EAR file for the application and a deployments subdirectory. The deployments subdirectory contains these files and subdirectories:
  - deployment.xml
    The file that contains configuration data on the application deployment.
  - META-INF
    The subdirectory that contains a J2EE application deployment descriptor file as well as IBM deployment extensions files and bindings files.
  - Subdirectories for all WAR and JAR files in the application. These subdirectories contain the XML and XMI configuration files for the web modules and enterprise bean modules that are included in the application.
- **/config/cells/*cellname*/nodes**
  The nodes subdirectory contains a subdirectory for the instance node.
- **/config/cells/*cellname*/nodes/*nodename***
  Node names for instances of WebSphere Application Server - Express are *hostname_instance*, where *hostname* is the iSeries system host name, and *instance* is the name of the instance.

  Each node subdirectory contains these files:

  Note: Some of these files have the same names as those in the containing cell directory. The configurations specified in node-level documents override the configurations specified in cell documents having the same name.
  - app.policy
    Contains default security permissions for application code.
  - library.policy
    Contains security permissions for shared libraries.
  - namestore.xml
    Contains persistent name binding data for the naming service.
  - node.xml
    Contains the configuration data for the node.
  - resources.xml
    Defines the resources that enterprise applications use. Resources include JDBC providers, data sources, and mail providers.
  - serverindex.xml
    Specifies the TCP/IP ports for special endpoints for each server under the node. Port values of services such as the naming service, SOAP service, and security services are specified here. This file also contains information on which enterprise applications are installed on each server.

- spi.policy
  Contains security permissions for service provider libraries such as resource providers.
- variables.xml
  Contains configuration variables used to specify directory paths. The variables can then be substitiuted for the actual path when specifying locations for WebSphere log files, JDBC implmentation classes, and application install paths.

• **/config/cells/*cellname*/nodes/*nodename*/servers/*servername***
  The name of the subdirectory corresponds to the name of the server. The name of the application server is the same as the name of your instance.

  The server directory contains a server.xml file, which provides configuration data specific to the server process. The server can also have files such as variables.xml and resources.xml, which provide additional configuration data that applies only to the server process.

  The server subdirectory can contain any of these files:

  Note: Some of these files have the same names as those in the containing node or cell directory. The configurations specified in server-level documents override the configurations specified in node and cell documents having the same name.

  - namestore-cell.xml
    Contains persistent cell-level name binding data for the naming service.
  - namestore-node.xml
    Contains persistent node-level name binding data for the naming service.
  - resources.xml
    Defines the resources that enterprise applications use. Resources include JDBC providers, data sources, and mail providers.
  - server.xml
    Contains configuration data for the services and components that run in the server process.
  - variables.xml
    Contains configuration variables used to specify directory paths. The variables can then be substitiuted for the actual path when specifying locations for WebSphere log files, JDBC implementation classes, and application install paths.

• **config/templates**
  The templates directory contains two subdirectories, named default and system, which contain template XML files for several configuration object types such as servers and JDBC providers.

  The WebSphere administrative console uses these templates when displaying default properties for a resource you are creating. You can also use these templates with the wsadmin scripting tool to create new resources based on the templates.

Here is an example of the structure.

```
config
  cells
    cell1
       cell.xml resources.xml virtualhosts.xml variables.xml security.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
       nodes
          nodeX
             node.xml variables.xml resources.xml serverindex.xml
             servers
               serverA
                  server.xml variables.xml
```

Properties files

Each instance of WebSphere Application Server - Express contains several properties files. The files in the properties directory for an instance are not managed by the administrative repository and file replication services. The properties files are located in the /QIBM/UserData/WebASE/ASE5/*instance*/properties directory, where *instance* is the name of your instance.

-
- client.policy
  The client.policy file is a default policy file shared by all of the WebSphere client containers and applets on a node.
- client_types.xml
  The client_types.xml file provides client type detection support for servlets extending PageListServlet. Using the configuration data in the client_types.xml file, servlets can determine the language type that calling clients require for the response.
- converter.properties
  The converter.properties file is used by the Web container to map an unsupported character set to a supported character set.
- encoding.properties
  The encoding.properties file is used by the Web container to map the language identifier to a character set.. This file is used only if the Web container determines the locale for a request using the Accept-Language HTTP header.
- ffdcRun.properties
  Properties file used to specify settings for the WebSphere Application Server - Express First Failure Data Capture (FFDC) diagnostic engine. This file should only be changed when working with IBM Service personnel to do problem determination.
- ffdcStart.properties
  Properties file used to specify settings for starting the WebSphere Application Server - Express First Failure Data Capture (FFDC) diagnostic engine. This file should only be changed when working with IBM Service personnel to do problem determination.
- ffdcStop.properties
  Properties file used to specify settings for stopping the WebSphere Application Server - Express First Failure Data Capture (FFDC) diagnostic engine. This file should only be changed when working with IBM Service personnel to do problem determination.
- implfactory.properties
  File for specifying implementation classes for WebSphere Application Server - Express runtime factories. This file should only be modified under the direction of IBM Service personnel.
- java.security
  The java.security file is used to specify various security properties for use by the java.security classes. WebSphere Application Server - Express uses this file instead of the java.security file located in the /QIBM/ProdData/Java400/jdk13/lib/security directory. If you wish to add security properties for your WebSphere Application Server - Express instance, you should modify this file in the properties directory for your instance.
- jmx.properties
  This is a configuration file that controls logging within the JMX environment. This file should only be modified under the direction of IBM Service personnel.
- samples.properties
  This file provides a list of the samples that are installed on your application server.
- sas.client.props
  The sas.client.props file contains the configuration settings for authentication between a Java client and a server.
- sas.server.props
  In WebSphere Application Server versions 4.0 and earlier, this file contained configuration settings for all application servers in an instance. In WebSphere Application Server - Express, these properties are

specified in the security.xml file. The sas.server.props file maps the values from previous versions to the XML values specified in the current version’s security.xml file.

- sas.stdclient.properties
  This file contains the default configuration settings for a secure Java client that requires a userid and password via standard input from the command line.
- sas.tools.properties
  This file contains the default configuration settings for a secure Java client that requires a userid and password via standard input from the command line.
- server.policy
  The server.policy file is a default policy file shared by all of the application servers on a node. The permissions in this file only pertain to the WebSphere Application Server - Express runtime code.
- soap.client.props
  This file contains configuration settings for authentication between a SOAP client and a server.
- sslbitsizes.properties
  This file contains properties associating SSL cipher suites to their bit sizes.
- TraceSettings.properties
  The TraceSettings.properties file is used to specify trace settings for client applications.
- was.policy
  Sometimes an application requires additional authentication information that is not specified in the app.policy file. The additional information is specified in the was.policy file.
- wsadmin.properties
  Contains properties used by the “The wsadmin administrative tool” on page 96.
- wsjaas.conf
  Contains configuration settings for the Java Authentication and Authorization Service (JAAS).
- wsjaas_client.conf
  Contains configuration settings for the Java Authentication and Authorization Service (JAAS) for client applications.
- wsserver.key
  Contains example configuration settings for allowing an application server to log on and access resources on a secure z/OS server.

WebSphere Application Server - Express default port definitions

This list includes the name of each port, the default value assigned to the port on pre-loaded systems, the configuration file where the port is specified, and a description that contains a link to documentation about how to customize the port setting.

Notes:

- For more information about port numbers that your iSeries system currently uses, enter the NETSTAT *CNN command on the CL command line. Press F14 to view assigned port numbers.
- You can also use the port validator tool to find port conflicts between different WebSphere Application Server instances, products, and servers. For information about the tool, see The port validator tool.
- **Web container port (HTTP_TRANSPORT)**
  - Default on pre-loaded systems: 2020
  - Configuration file: server.xml, plugin-cfg.xml, virtualhosts.xml
  - Description: The TCP/IP port on which the Web container listens for requests from the Web server. You can specify this port with the WebSphere administrative console HTTP transort settings page or with “Change application server ports with the chgwassvr script” on page 50. If you change this port, you must “Regenerate the Web server plugin configuration” on page 41 for the application server.
- **Web container secure port (HTTPS_TRANSPORT)**
  - Default on pre-loaded systems: no default value

  - Configuration file: server.xml, plugin-cfg.xml, virtualhosts.xml
  - Description: The TCP/IP port on which the Web container listens for secure requests from the Web server. You can specify this port with the WebSphere administrative console HTTP transort settings page or with “Change application server ports with the chgwassvr script” on page 50. If you change this port number, remember the following information:
    - To use secure (SSL enabled) ports you must have the i5/OS Digital Certificate Manager product (5722SS1 option 34) and a Cryptographic Access Provider product (such as 5722AC3) installed. For more information see Configure SSL.
    - If you change this port, you must “Regenerate the Web server plugin configuration” on page 41 for the application server.
- **WebSphere administrative console port (HTTP_TRANSPORT_ADMIN)**
  - Default on pre-loaded systems: 2039
  - Configuration file: server.xml, virtualhosts.xml
  - Description: The TCP/IP port on which the Web container listens for requests for the administrative application. You can specify this port with the WebSphere administrative console HTTP transort settings page or with “Change application server ports with the chgwassvr script” on page 50.
- **WebSphere administrative console secure port (HTTPS_TRANSPORT_ADMIN)**
  - Default on pre-loaded systems: 2040
  - Configuration file: server.xml, virtualhosts.xml
  - Description: The TCP/IP port on which the Web container listens for secure requests for the administrative application. You can specify this port with the WebSphere administrative console HTTP transort settings page or with “Change application server ports with the chgwassvr script” on page 50.
- **Name service or RMI connector port (BOOTSTRAP_ADDRESS)**
  - Default on pre-loaded systems: 2030
  - Configuration file: serverindex.xml
  - Description: The TCP/IP port on which the name service listens. This port is also the RMI connector port. Specify this port with the WebSphere administrative console End point settings page or with “Change application server ports with the chgwassvr script” on page 50.
- **Simple Object Access Protocol (SOAP) port (SOAP_CONNECTOR_ADDRESS)**
  - Default on pre-loaded systems: 2035
  - Configuration file: serverindex.xml
  - Description: The TCP/IP port that your server uses for Simple Object Access Protocol (SOAP). Specify this port with the WebSphere administrative console End point settings page or with “Change application server ports with the chgwassvr script” on page 50.
- **Secure Association Services (SAS)**
  - Default on pre-loaded systems: 1036
  - Configuration file: serverindex.xml
  - Description: The port on which the Secure Association Services (SAS) listen for inbound authentication requests. Specify this port with the WebSphere administrative console End point settinga page or with “Change application server ports with the chgwassvr script” on page 50.
- **Common Secure Interoperability Version 2 (CSIV2) Mutual**
  - Default on pre-loaded systems: 2037
  - Configuration file: serverindex.xml
  - Description: The port on which the CSIV2 Service listens for inbound client authentication requests. Specify this port with the WebSphere administrative console End point settings page or with the “Change application server ports with the chgwassvr script” on page 50.
- **CSIV2 Server**
  - Default on pre-loaded systems: 2038

– Configuration file: serverindex.xml
– Description: The port on which the CSIV2 Service listens for inbound server authentication requests. Specify this port with the WebSphere administrative console End point settings page or with the "Change application server ports with the chgwassvr script" on page 50.

User profiles and authorities

WebSphere Application Server - Express uses two i5/OS user profiles by default:

- QEJB
- QEJBSVR

The QEJB user profile is is shipped as part of the i5/OS operating system. This user profile is used only when accessing validation list objects used for storing the encoded passwords used with WebSphere Application Server - Express. For more information on using validation list objects to store encoded passwords, see Password encoding

The QEJBSVR user profile is created on your iSeries when you install WebSphere Application Server - Express. This profile is the default profile under which all application servers run. Directories and files used by WebSphere Application Server - Express are normally owned by user profile QEJBSVR. The WebSphere Application Server - Express runtime, administration tools, and Qshell scripts sets the ownership and authorities correctly on any objects created. If you create objects manually outside of the WebSphere Application Server - Express tools, or if you modify the authorities on objects used by WebSphere Application Server - Express, you must ensure QEJBSVR has the correct authorities to these objects.

When you create new directories for WebSphere Application Server - Express, the QEJBSVR user profile must have read and execute authorities (*RX) to those directories.

Note: If you have specified another user profile to run your application servers, it is recommended that you specify QEJBSVR for its group profile. See Run under a different profile for more information.

Set the time zone

To ensure that the application server runtime and your application components have the correct date and time values, set the **user.timezone** property. The syntax of the property is

```
user.timezone=timezone
```

where *timezone* is the supported value for your time zone. For a list of supported values, see "Supported user.timezone property values for the Development Kit for Java(R) 1.3" on page 146.

Note: The Java(TM) virtual machine calculates the time based on the value of the user.timezone property and the system values QHOUR and QUTCOFFSET. QUTCOFFSET represents the number of hours difference between the system's time zone and Greenwich Mean Time (GMT). The Java virtual machine adds the values of QHOUR and QUTCOFFSET to calculate GMT, then uses GMT and value of the user.timezone property to derive the correct time of the day.

You can set the user.timezone property in several different files. The time zone setting has different effects based on the file in which it is specified.

- **Set the property to affect all Java virtual machine processes on your iSeries server.**
  Edit the user.timezone property in the /QIBM/UserData/Java400/SystemDefault.properties file. If the file does not exist, create it in this directory.
- **Set the property to affect application servers.**
  You have two options:

- Edit the /home/*user_ID*/SystemDefault.properties file, where *user_id* is the user profile under which the application server runs. By default, this is the QEJBSVR user profile. If the file does not exist, create it in this directory. If you specify the time zone in this file, only application server jobs are affected.
- In the administrative console, add the user.timezone property to the Java virtual machine system properties for your application server:
  1. "Start the WebSphere administrative console" on page 83.
  2. In the topology tree, expand **Servers** and click **Application Servers**.
  3. Click the name of the application server for which you want to set the time zone.
  4. On the application server page, click **Process Definition**.
  5. On the **Process Definition** page, click **Java Virtual Machine**.
  6. On the **Java Virtual Machine** page, click **Custom Properties**.
  7. On the **Custom Properties** page, click **New**.
  8. Specify user.timezone in the Name field and *timezone* in the Value field, where *timezone* is the supported value for your time zone.
  9. Click **Apply**.
  10. "Save the application server configuration" on page 84.

Configure a locale to specify the time zone

You can also configure your application server to run in a locale. The locale determines the time zone in which a Java virtual machine operates. To use locales, follow these steps:

1. If Extended NLS Support is not already installed on your iSeries server, install it by selecting option 21 when you install the i5/OS base operating system (5769-SS1).
2. Run the Create File (CRTF) command to create a locale source physical file from file LOCALSRC in library QSYSLOCALE.
3. Run the Start SEU (STRSEU) command to edit the source file.
4. Specify a time zone in the file.

   Note: 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 time zone field TNAME. The value of TNAME must match the name of a Java time zone.
5. Run the Create Locale (CRTLOCALE) command to create a locale from the source file.
6. Run the Change User Profile (CHGUSRPRF) command to change the user profile under which the application server runs. Edit the user profile to use the new locale.

If you use more than one method to specify the time zone, the application server prioritizes the methods in this order:

- Java virtual machine system property
- User directory SystemDefault.properties file
- java400$^{(TM)}$ SystemDefault.properties file
- Locale

Supported user.timezone property values for the Development Kit for Java(R) 1.3

The following table lists the values for the user.timezone property that WebSphere Application Server for iSeries supports for the Development Kit for Java(R) 1.3. On iSeries, the QTIMZON system variable specifies the time zone.

Notes:

- The **Time zone ID** column lists time zones (in **boldface**) and 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 **QTIMZON variable** column lists the corresponding value for the QTIMZON system variable. If multiple values are specified in this column, either value is acceptable.

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
MIT	-11 : 00		West Samoa Time	
Pacific/Apia	-11 : 00		West Samoa Time	QN1100UTCS
Pacific/Niue	-11 : 00		Niue Time	
Pacific/Pago_Pago	-11 : 00		Samoa Standard Time	
America/Adak	-10 : 00	60	Hawaii-Aleutian Standard Time	QN1000HAST
HST	-10 : 00		Hawaii Standard Time	
Pacific/Fakaofo	-10 : 00		Tokelau Time	
Pacific/Honolulu	-10 : 00		Hawaii Standard Time	QN1000UTCS
Pacific/Rarotonga	-10 : 00		Cook Is. Time	
Pacific/Tahiti	-10 : 00		Tahiti Time	
Pacific/Marquesas	-9 : 30		Marquesas Time	
AST	-9 : 00	60	Alaska Standard Time	QN0900AST
America/Anchorage	-9 : 00	60	Alaska Standard Time	
Pacific/Gambier	-9 : 00		Gambier Time	QN0900UTCS
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	
PST	-8 : 00	60	Pacific Standard Time	QN0800PST QN0800U
Pacific/Pitcairn	-8 : 00		Pitcairn Standard Time	QN0800UTCS
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/Mazatlan	-7 : 00	60	Mountain Standard Time	
America/Phoenix	-7 : 00		Mountain Standard Time	QN0700MST2 QN0700UTCS
MST	-7 : 00	60	Mountain Standard Time	QN0700MST QN0700T
PNT	-7 : 00		Mountain Standard Time	

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
America/Belize	-6 : 00		Central Standard Time	
America/Chicago	-6 : 00	60	Central Standard Time	
America/Costa_Rica	-6 : 00		Central Standard Time	QN0600UTCS
America/El_Salvador	-6 : 00		Central Standard Time	
America/Guatemala	-6 : 00		Central Standard Time	
America/Managua	-6 : 00		Central Standard Time	
America/Mexico_City	-6 : 00	60	Central Standard Time	
America/Regina	-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 QN0600S
Pacific/Easter	-6 : 00	60	Easter Is. Time	
Pacific/Galapagos	-6 : 00		Galapagos Time	
America/Bogota	-5 : 00		Colombia Time	
America/Cayman	-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/Indianapolis	-5 : 00		Eastern Standard Time	QN0500UTCS
America/Jamaica	-5 : 00		Eastern Standard Time	
America/Lima	-5 : 00		Peru 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/Panama	-5 : 00		Eastern Standard Time	

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
America/Port-au-Prince	-5 : 00		Eastern Standard Time	
America/Porto_Acre	-5 : 00		Acre Time	
America/Rio_Branco	-5 : 00		GMT-05:00	
EST	-5 : 00	60	Eastern Standard Time	QN0500EST
IET	-5 : 00		Eastern Standard Time	QN0500EST2
America/Anguilla	-4 : 00		Atlantic Standard Time	
America/Antigua	-4 : 00		Atlantic Standard Time	
America/Aruba	-4 : 00		Atlantic Standard Time	
America/Asuncion	-4 : 00	60	Paraguay Time	
America/Barbados	-4 : 00		Atlantic Standard Time	
America/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/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/Puerto_Rico	-4 : 00		Atlantic Standard Time	QN0400UTCS
America/Santiago	-4 : 00	60	Chile Time	
America/Santo_Domingo	-4 : 00		Atlantic Standard Time	

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
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	
Antarctica/Palmer	-4 : 00	60	Chile Time	
Atlantic/Bermuda	-4 : 00	60	Atlantic Standard Time	QN0400AST
Atlantic/Stanley	-4 : 00	60	Falkland Is. Time	
PRT	-4 : 00		Atlantic Standard Time	
America/St_Johns	-3 : 30	60	Newfoundland Standard Time	
CNT	-3 : 30	60	Newfoundland Standard Time	QN0330NST
AGT	-3 : 00		Argentine Time	
America/Buenos_Aires	-3 : 00		Argentine Time	QN0300UTCS
America/Cayenne	-3 : 00		French Guiana Time	
America/Fortaleza	-3 : 00		Brazil Time	
America/Godthab	-3 : 00	60	Western Greenland Time	
America/Miquelon	-3 : 00	60	Pierre & Miquelon Standard Time	
America/Montevideo	-3 : 00		Uruguay Time	
America/Paramaribo	-3 : 00		Suriname Time	
America/Sao_Paulo	-3 : 00	60	Brazil Time	
BET	-3 : 00	60	Brazil Time	QN0300UTC2
America/Noronha	-2 : 00		Fernando de Noronha Time	QN0200UTCS
Atlantic/South_Georgia	-2 : 00		South Georgia Standard Time	
America/Scoresbysund	-1 : 00	60	Eastern Greenland Time	
Atlantic/Azores	-1 : 00	60	Azores Time	
Atlantic/Cape_Verde	-1 : 00		Cape Verde Time	QN0100UTCS
Atlantic/Jan_Mayen	-1 : 00		Eastern Greenland Time	

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
Africa/Abidjan	0 : 00		Greenwich Mean Time	
Africa/Accra	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/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	
Atlantic/Canary	0 : 00	60	Western European Time	
Atlantic/Faeroe	0 : 00	60	Western European Time	
Atlantic/Reykjavik	0 : 00		Greenwich Mean Time	
Atlantic/St_Helena	0 : 00		Greenwich Mean Time	
Europe/Dublin	0 : 00	60	Greenwich Mean Time	
Europe/Lisbon	0 : 00	60	Western European Time	
Europe/London	0 : 00	60	Greenwich Mean Time	Q0000GMT2
GMT	0 : 00		Greenwich Mean Time	Q0000GMT
UTC	0 : 00		Coordinated Universal Time	Q0000UTC

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
WET	0 : 00	60	Western European Time	
Africa/Algiers	1 : 00		Central European Time	QP0100CET QP0100UTCS
Africa/Bangui	1 : 00		Western African 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	
ECT	1 : 00	60	Central European Time	QP0100CET3
Europe/Amsterdam	1 : 00	60	Central European Time	
Europe/Andorra	1 : 00	60	Central European Time	
Europe/Belgrade	1 : 00	60	Central European Time	
Europe/Berlin	1 : 00	60	Central European Time	
Europe/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/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	

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
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/Stockholm	1 : 00	60	Central European Time	
Europe/Tirane	1 : 00	60	Central European Time	
Europe/Vaduz	1 : 00	60	Central European Time	
Europe/Vienna	1 : 00	60	Central European Time	
Europe/Warsaw	1 : 00	60	Central European Time	
Europe/Zurich	1 : 00	60	Central European Time	QP0100CET2
ART	2 : 00	60	Eastern European Time	
Africa/Blantyre	2 : 00		Central African Time	
Africa/Bujumbura	2 : 00		Central African Time	
Africa/Cairo	2 : 00	60	Eastern European Time	
Africa/Gaborone	2 : 00		Central African Time	
Africa/Harare	2 : 00		Central African Time	
Africa/Johannesburg	2 : 00		South Africa Standard Time	QP0200SAST
Africa/Kigali	2 : 00		Central African Time	
Africa/Lubumbashi	2 : 00		Central African Time	
Africa/Lusaka	2 : 00		Central African Time	
Africa/Maputo	2 : 00		Central African Time	
Africa/Maseru	2 : 00		South Africa Standard Time	
Africa/Mbabane	2 : 00		South Africa Standard Time	
Africa/Tripoli	2 : 00		Eastern European Time	
Asia/Amman	2 : 00	60	Eastern European Time	
Asia/Beirut	2 : 00	60	Eastern European Time	
Asia/Damascus	2 : 00	60	Eastern European Time	
Asia/Jerusalem	2 : 00	60	Israel Standard Time	

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
Asia/Nicosia	2 : 00	60	Eastern European Time	
CAT	2 : 00		Central African Time	
EET	2 : 00	60	Eastern European Time	QP0200EET
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/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		Eastern European Time	QP0200EET2 QP0200UTCS
Europe/Vilnius	2 : 00		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	
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	

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
Asia/Riyadh	3 : 00		Arabia Standard Time	
EAT	3 : 00		Eastern African Time	
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	
Asia/Tehran	3 : 30	60	Iran Time	
MET	3 : 30	60	Iran 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/Tbilisi	4 : 00	60	Georgia Time	
Asia/Yerevan	4 : 00	60	Armenia Time	
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/Tashkent	5 : 00		Uzbekistan Time	
Asia/Yekaterinburg	5 : 00	60	Yekaterinburg Time	
Indian/Chagos	5 : 00		Indian Ocean Territory Time	
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	
Asia/Almaty	6 : 00	60	Alma-Ata Time	QP0600UTC2
Asia/Colombo	6 : 00		Sri Lanka Time	

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
Asia/Dacca	6 : 00		Bangladesh Time	
Asia/Dhaka	6 : 00		Bangladesh Time	QP0600UTCS
Asia/Novosibirsk	6 : 00	60	Novosibirsk Time	
Asia/Thimbu	6 : 00		Bhutan Time	
Asia/Thimphu	6 : 00		Bhutan Time	
BST	6 : 00		Bangladesh Time	
Asia/Rangoon	6 : 30		Myanmar Time	
Indian/Cocos	6 : 30		Cocos Islands Time	
Asia/Bangkok	7 : 00		Indochina Time	
Asia/Jakarta	7 : 00		Java Time	QP0700WIB
Asia/Krasnoyarsk	7 : 00	60	Krasnoyarsk Time	
Asia/Phnom_Penh	7 : 00		Indochina Time	
Asia/Saigon	7 : 00		Indochina Time	QP0700UTCS
Asia/Vientiane	7 : 00		Indochina Time	
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/Hong_Kong	8 : 00		Hong Kong Time	QP0800JIST QP0800UTCS
Asia/Irkutsk	8 : 00	60	Irkutsk Time	
Asia/Kuala_Lumpur	8 : 00		Malaysia Time	
Asia/Macao	8 : 00		China Standard 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		Borneo Time	QP0800WITA
Asia/Ulaanbaatar	8 : 00		Ulaanbaatar Time	
Asia/Ulan_Bator	8 : 00		Ulaanbaatar Time	
Australia/Perth	8 : 00		Western Standard Time (Australia)	QP0800AWST
CTT	8 : 00		China Standard Time	QP0800BST
Asia/Jayapura	9 : 00		Jayapura 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	
JST	9 : 00		Japan Standard Time	QP0900JST

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
Pacific/Palau	9 : 00		Palau 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)	
AET	10 : 00	60	Eastern Standard Time (New South Wales)	QP1000AEST
Antarctica/ DumontDUrville	10 : 00		Dumont-d'Urville Time	
Asia/Vladivostok	10 : 00	60	Vladivostok Time	
Australia/Brisbane	10 : 00		Eastern Standard Time (Queensland)	
Australia/Hobart	10 : 00	60	Eastern Standard Time (Tasmania)	
Australia/Sydney	10 : 00	60	Eastern Standard Time (New South Wales)	
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	
Australia/ Lord_Howe	10 : 30	30	Load Howe Standard Time	
Asia/Magadan	11 : 00	60	Magadan Time	
Pacific/Efate	11 : 00		Vanuatu Time	
Pacific/Guadalcanal	11 : 00		Solomon Is. Time	QP1100UTCS
Pacific/Kosrae	11 : 00		Kosrae Time	
Pacific/Noumea	11 : 00		New Caledonia Time	
Pacific/Ponape	11 : 00		Ponape Time	
SST	11 : 00		Solomon Is. Time	
Pacific/Norfolk	11 : 30		Norfolk Time	
Antarctica/McMurdo	12 : 00	60	New Zealand Standard Time	

Time zone ID	Raw offset Hours : Minutes	DST offset Minutes	Display name	QTIMZON variable
Asia/Anadyr	12 : 00	60	Anadyr Time	
Asia/Kamchatka	12 : 00	60	Petropavlovsk-Kamchatski Time	
NST	12 : 00	60	New Zealand Standard Time	QP1200NZST
Pacific/Auckland	12 : 00	60	New Zealand Standard Time	
Pacific/Fiji	12 : 00		Fiji Time	QN1200UTCS QP1200UTCS
Pacific/Funafuti	12 : 00		Tuvalu 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	
Pacific/Chatham	12 : 45	60	Chatham Standard Time	QP1245UTCS
Pacific/Enderbury	13 : 00		Phoenix Is. Time	
Pacific/Tongatapu	13 : 00		Tonga Time	
Pacific/Kiritimati	14 : 00		Line Is. Time	

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY  10504-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
```

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

Programming Interface Information

This WebSphere Application Server - Express publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

Terms and conditions

Printed in USA