

Migration -- table of contents

3: Migration overview

3.0: Transitioning to Version 4.0

3.1: Migrating product prerequisites

3.2: Migrating from previous product versions

3.2.1: Migration from Version 2.0x

3.2.1.1: Migration from Version 2.0x to Version 3.0

3.2.2: Migrating from Version 3.x

3.2.2.1: Using automated migration support

3.2.2.2: Migrating configurations manually

3.2.2.2.1: Saving the current configuration by using migration tools

3.2.2.2.2: Saving the current configuration manually

3.2.2.2.3: Restoring the previous configuration to the new installation

3.2.2.2.4: Mapping configurations to Version 4.0

3.2.3: Migrating from Version 4.0 Advanced Single Server Edition

3.2.4: Migrating Web server plug-ins

3.3: Migrating APIs and specifications

3.3.1: Migrating to supported EJB specification

3.3.2: Migrating to supported Servlet specification and extensions

3.3.2.1: Example: Migrating `HttpServletResponse.callPage()`

3.3.4: Migrating to supported XML API

3.3.5: Migrating to supported user profile APIs

3.3.6: Migrating session management

3.3.6.1: Migrating from Version 2.0 session support

3.3.7: Migrating to supported security APIs

3.3.8: Migrating to supported database connection APIs (and JDBC)

3.3.8.1: Migrating from the Version 3.0x connection pooling model

3.3.8.2: Migrating servlets from the connection manager model

3.3.8.3: Obsolete connection manager APIs

3.3.9: Migrating to supported transaction support

3.3.10: Migrating to supported XML configuration

3.5: Switching administrative databases

3.5.1: Using the database conversion assistant to switch administrative databases

3.5.2: Settings to change when switching administrative databases

3.5.3: Switching server databases to DB2/390: Switching administrative databases

3.6: Interoperability with z/OS

3.7: Interoperability with Version 3.5.x

3: Migration overview

Migration focuses on leveraging the existing environment and applications, changing them to be compatible with the current product version, instead of starting from the beginning.

Migration for IBM WebSphere Application Server Version 4.0 includes the following activities:

Activity	Where to find instructions
<p>1. Migrate or upgrade product prerequisites to supported versions</p> <p>As the product version changes, its prerequisites or corequisites also change. It is probably necessary to update your database, Webserver, JDK version, and other software.</p>	<p>Article 3.1</p>
<p>2. Migrate or upgrade to IBM WebSphere Application Server Version 4.0</p> <p>In most cases, migration programs are available to ease the transition. However, some manual preparation may be necessary.</p> <p>Programmatic support for migration from Version 2.0x is not provided. To migrate your installation from Version 2.0x, follow the documentation, starting at article 3.2.1.</p>	<p>Article 3.2. See also Installing the product</p>
<p>3. Migrate administrative configurations</p> <p>If your company has been using a previous product version, the system administrator has probably fine-tuned various application and server settings for the environment. It is important to have a strategy for migrating these settings with maximum efficiency and minimal loss.</p>	<p>Start with Article 3.2.2.</p>
<p>4. Update application code to supported specification and API levels</p> <p>Note: Support for several APIs has been removed in Version 4.0.</p> <p>Section 4 of the InfoCenter focuses on developing new applications, though it also outlines new APIs whose functions you might add to existing applications in a piecemeal fashion.</p>	<p>Article 3.3</p>
<p>5. Redeploy applications on Version 4.0</p>	<p>Article 6.3.2</p>

Before you begin, be sure to read [Transitioning to Version 4.0](#). That article outlines the reorganization of the Version 4.0 product relative to Version 3.x.

Transitioning to Version 4.0

This article is written for users of IBM WebSphere Application Server Version 3.x who are upgrading to Version 4.0. Version 4.0 is fully compliant with Java 2 Platform, Enterprise Edition (J2EE) specifications, which has caused many changes in the organization of the product relative to that of Version 3.x.

J2EE creates a clear separation between development (creating the application) and administration (installing and managing the application). This separation enables the development of applications that are independent from the environments in which they are deployed. In addition, J2EE task separation simplifies the process of promoting an application from initial development up through production, or of moving an application from one server to another. In each of these cases, changes to application code are not necessary; only deployment parameters might change.

Version 4.0 supports J2EE task separation through reorganized interfaces. In Version 3.x, developers used the administrative console to create, edit, and view applications. In Version 4.0, developers use the *Application Assembly Tool* (AAT) to create, edit, and view J2EE applications.

In Version 4.0, each application is installed into the server domain and bound to an environment when the application is installed. This enables administration at the application and module level. Administrators no longer need to manage individual servlets, JSPs, or beans.

The relationship between applications and application servers has changed in J2EE. An enterprise application can contain many Web modules and EJB modules. Each module can be installed onto a different application server or server group, even if the servers or server groups are on multiple nodes. As a result, a single application can contain many modules, spread over many application servers or server groups. Likewise, a single application server or server group can have modules from many different applications installed on it.

After a J2EE application is created, you install it onto application servers through the administrative console. Through the administrative console, you can view installed modules either by the application to which they belong or by the application server on which they are installed. Modules can be started and stopped individually as well as collectively. Modules can be started collectively by either starting the application to which they belong or starting the application server on which they are installed. Modules can be stopped in a similar way.

Deploying new J2EE applications

There are two steps involved in creating J2EE applications: copying the appropriate files into the archive (classes, JSPs, HTML, image files, and so on) and creating deployment descriptor files for the modules and applications. In Version 4.0, the AAT supports both steps by enabling users to copy files with appropriate relative paths into the archive, as well as by providing a GUI method for defining deployment descriptors.

Developers can also set environment-specific binding information through the AAT. These bindings are used as defaults when the application is installed through the administrative console. In addition, users can define IBM extensions to the J2EE specification, such as allowing servlets to be served by class name. To ensure portability to other application servers, these extensions are saved in a separate XML file from the standard J2EE deployment descriptor.

Role-based security

Version 4.0 security is consistent with J2EE role-based security specifications. Roles are specified in the deployment descriptors for an application; these roles are then bound to users or to groups when the application is installed. In the administrative console, a Security Center enables you to perform all security tasks from a single location. This encompasses everything from changing the binding information for roles in an application to setting SSL properties to enabling security. Application-specific security tasks can also be done through the

property sheets for each application.

Redeployment of previously installed applications

In Version 3.x, all tasks were performed through the administrative console. In Version 4.0, application settings are defined in J2EE deployment descriptors through the AAT.

Unless you must change information that affects the bindings of an installed application, you can edit and save the deployment descriptors in place. To redeploy such an application, open the AAT directly on the installedApps folder that holds the application.

You can also create or edit applications manually. For example, if you need to add a JSP or change a servlet class, you can simply place the new or changed file in the appropriate location in the installedApps folder.

To redeploy an installed application that requires changes to binding, you export the application through the administrative console, edit the application in the AAT, and reinstall the application through the administrative console. Because existing binding information is saved during the export step, the only additional binding information needed is for the components or modules added during editing.

Important: For security and consistency, Web application URLs are now case-sensitive on all operating systems. For details, see the related information.

Support for J2EE resource types

In addition to JDBC providers and datasources, several resource types were added in Version 4.0: URLs, JMS, and JavaMail. In each case, you can create a resource provider (JDBC providers, URL providers, and JMS providers) and then create resource factories for each provider (datasources, URLs, JavaMail sessions, JMS destinations, and JMS connections). In the case of JavaMail, the default JavaMail provider is not shown in the administrative console, because it is not configurable and additional JavaMail providers cannot be created.

J2EE impact to models and cloning

In Version 3.x, many different types of objects can be modeled and cloned. With the switch to J2EE compliance in Version 4.0, only application servers can be cloned. These models are now called *server groups*, and each server group can contain multiple application servers, or *clones*.

Where to go next for more information

For more information about J2EE, visit the following Web site:

<http://java.sun.com>

For more information about changes in configuration support, see the related information. For information about how to upgrade to Version 4.0, see [Migration](#).

3.1: Migrating product prerequisites

The prerequisites Web page described in [article 1.3](#) contains up-to-date information about the supported prerequisites and corequisites.

Be sure to check whether your JDBC provider is at the right level for the new installation. This driver will be required by the product administrative server in order to connect to its administrative database.

Migrating DB2, IBM HTTP Server, and other complimentary prerequisites

IBM WebSphere Application Server simplifies the migration of product prerequisites by providing the option to install a complimentary Web server, database, and JDK on your supported operating system. The JDK is the exact level and type needed by IBM WebSphere Application Server. See the installation guides for further details.

The compact disc version of the product includes the complimentary prerequisites; Web download versions can vary (offered with and without database, and so on), to provide a choice of download file sizes. If not installing from CD, consult the product Web site for details. Make sure you download the installation package with the features you want.

You can uninstall the back-level prerequisites and install brand-new versions when you install the product.

Migrating non-IBM prerequisites

Some prerequisite or corequisite products, such as an Oracle or Sybase database, are not provided as part of the IBM WebSphere Application Server installation. To upgrade these, the best source of information is the documentation for the products.

First, consult the previously cited prerequisites page to determine which software requires migration or upgrade. Second, consult the documentation for the particular products to learn how to migrate to the version supported by this product.

For prerequisites not offered during the Application Server installation, the safest approach is to migrate or upgrade prerequisites *before* installing IBM WebSphere Application Server.

3.2: Migrating from previous product versions

Programmatic support for migration from Version 2.0x is not provided. To migrate your installation from Version 2.0x, follow the documentation, starting at article 3.2.1.

Migration from Version 3.x is part of the Version 4.0 installation program. All 3.x releases are supported except for Version 3.5.0. For an overview of the process, see article 3.2.2.

You can also migrate from Version 4.0 Advanced Single Server Edition. As with Version 3.x migration, the installation program can do much of this for you. For information about how to do this manually, see article 3.2.3.

3.2.1: Migrating from Version 2.0x

Two main paths are available for migrating from Version 2.0x.

Version 2.0x > Version 3.0x > Version 3.5 or higher

IBM WebSphere Application Server provides automatic migration from versions 3.02 or later. Migrating to Version 3.02 from versions 3.0 or 3.01 is fairly trivial based on upgrade options added to the Application Server installation program. For documentation about migrating from Version 2.0x to Version 3.0x, see the Related information.

Install product as new and migrate files and settings by hand

Uninstall Version 2.0x and start new with this version, transferring application files and configuration settings by hand.

Earlier versions of this product differ dramatically in terms of supported programming specifications, file placement, and administrative settings. In the absence of a comprehensive automated migration tool from Version 2.0x to Version 3.0x, the effort required to migrate to this version by way of Version 3.0x varies little from the effort required to install the product from scratch.

3.2.1.1: Migration from Version 2.0x to Version 3.0

This article is for Version 2.0x users who have chosen to migrate to Version 3.5 or later by way of Version 3.0. After you have upgraded your Version 2.0x installation to Version 3.0 as specified in these instructions, install PTF 2 from the product Web site. At that point, you can use automated migration support to upgrade the product to Version 3.5 or later.

For complete Version 3.0x installation and configuration information, consult the product Web site Library page cited in the Related information.

Preparation before installing Version 3.0x

Before uninstalling any previous version of the product, be sure the files that you want to migrate will be saved. The graphical user interface displayed when you install Version 3.0x backs up the files in the following directories:

1. classes
2. realms
3. servlets
4. properties, including the files--
 - o servlet.properties
 - o admin_port.properties
 - o rules.properties
 - o jvm.properties
 - o aliases.properties
 - o commgr.properties
 - o userprofile.properties

If you have files that reside outside of those four directories (for example, if you created your own directory in the product installation), back up the files in a location outside of the current installation before installing Version 3.0x.

Before uninstalling Version 2.0x, back up files and directories so that you can perform the following procedure after installing Version 3.0x:

1. Copy the Version 2.0x servlets directory to the Version 3.0x directory ...\\WebSphere\\AppServer\\hosts\\default_host\\default_app\\servlets.
2. Copy all files in the Version 2.0x \\classes directory to the Version 3.0x \\classes directory.
3. Copy all files in the Version 2.0x \\web\\classes directory to the Version 3.0x \\web\\classes directory.

Additional work after installing Version 3.0x

Before uninstalling Version 2.0x, you backed up some files in preparation for the previous steps. Finish those steps now.

Next, migrate the Enterprise Java Services server. Run the Enterprise Java Services migration program (*.exe file) to migrate the Enterprise Java Services server from Version 2.0x to Version 3.0x. The program will parse the Version 3.0x ejb.properties file. It will also update the appropriate entries in the Enterprise Java Services server database using the administration server command line. This script file is included with the installation program.

Enterprise Java Services require that the database be connected to complete the migration process.

Migrating administrative data

To assist you in moving administrative data from Version 2.0x to Version 3.0x, you can use a migration tool developed to move the data.

To start the data migration tool, use the following command:

```
java com.ibm.ejs.sm.ejscp.scripts.Migrate -file properties_file
-node node_name -jarFile DB2_driver_jarfile
[-trace]
```

properties_file is the name of the configuration file. node_name is the name of the node. DB2_driver_jarfile is the name of the jar/zip file containing the JDBC provider. -trace enables tracing.

Running the tool gives you an ejscp script as the output. The output file name is Upgradenode_name.tcl. To complete the migration, run the script using ejscp with the following command:

```
java tcl.lang.Script Upgradenode_name.tcl
```

3.2.2: Migrating from Version 3.x

Before you start, be sure to read [Transitioning to Version 4.0](#). That article outlines the reorganization of the Version 4.0 product relative to Version 3.x.

A summary of the product migration process follows. Most of this can be done for you by the product installation program.

1. Back up the current administrative configuration and user data files in the current installation root directory. See 3.2.2.2.1 for more information.
2. Stop and uninstall the current version of IBM WebSphere Application Server.
3. Drop the administrative database used for the previous installation (for example, the WAS database)
 1. Back up the database. You can use the database management system or simply rename the database.
 2. Drop the database.
4. [Install the new version of IBM WebSphere Application Server](#).
5. Start the new administrative server.
6. Restore the configuration in the new installation.

Migrating administrative configurations

Tools for migrating administrative configurations are provided for versions 3.02 and later. This support enables either edition of Version 3.x to be upgraded to either Advanced or Advanced Single Server edition of Version 4.0. There are two ways to migrate from Version 3.x:

- Use the automated migration that is part of product installation.
- Manually complete the same steps as the automated migration support would. This might be necessary for nonstandard installations.

For details, see the [Related information](#).

3.2.2.1: Using automated migration support

In IBM WebSphere Application Server Version 4.0, automated migration support is part of the installation program. In the preinstallation phase, the installation program detects previously installed versions, collects information about how you want the migrated installation to look, and exports the current administrative configuration. After you have updated product prerequisites, you run the second phase, which installs the new version of the product and imports the backed-up administrative configuration.

Before you begin, make sure the administrative server is running.

Preinstallation phase

1. When you start the installation program, it automatically detects previous installed versions of the product and displays them in a list. If the installation program supports migration from a selected version, a **Perform migration** check box appears above the list.
2. The installation program prompts you for the following information:
 - **[Windows NT and Windows 2000 only]** Directory for the new installation (default is the current installation root). Be sure this is correct: This information is not used until the next phase, but you will not be able to modify it then.
 - Backup directory.
 - Directory for temporary staging.
 - Directory for migration log.
3. The installation program prompts you to start the administrative server.
4. Click **Start Migration**. The installation program exports the current configuration and displays the migration log file.
 - On UNIX and Linux installations, the migration log is displayed only after migration is finished.
 - On Windows NT and Windows 2000 installations, display of the migration log file is refreshed throughout the migration process.
5. When the installation program prompts you to check the migration log file, do *one* of the following:
 - If the migration log file indicates success, click **Finish**. The next time you run the installation program, the next phase will start.
 - Otherwise, click **Cancel** and correct the logged errors. The next time you run the installation program, migration will start from the beginning.

Note for UNIX users: If the migration log file indicates problems with migration and you click **Finish**, you cannot rerun this phase of migration until you delete the file `/tmp/WAS_Migration_temp.properties`.

When this phase finishes successfully, do the following:

1. [Migrate prerequisites](#).
2. If you are installing Version 4.0 into the same directory structure as Version 3.x, do *one* of the following:
 - If the migration backup directory is within the Version 3.x directory structure, retain the migration backup directory but delete the rest of the Version 3.x directory structure.
 - Otherwise, delete the entire Version 3.x directory structure.
3. Restart the installation program.

Installation and postinstallation phase

1. When you restart the product installation program, it detects the following problems:
 - Prerequisites that have not been upgraded appropriately.
 - A running application server, if the new installation directory is the same as the current one. The product must be stopped in order for the installation program to properly overlay the files.
2. The installation program takes you through [the standard installation process](#).

[Windows NT and Windows 2000 only] Unlike during the standard installation process, however, you cannot modify the installation directory during the combined installation and migration.

3. After installation is finished, postinstallation migration is performed. The installation program imports the configuration that was exported in the first phase and displays the migration log file.

If this phase finishes successfully, restart the administrative server in order to complete the setup. The administration server takes longer to start up this first time, because it is loading the configuration file that was created during the migration process.

3.2.2.2: Migrating configurations manually

Manual migration might be necessary if either your current installation or your Version 4.0 installation requirements vary too much from assumptions made by the product installation program. This article outlines the first and last steps of the overall product migration process, as follows:

- Before upgrading the product, save the current configuration and back up necessary files.
- After upgrading the product, [restore the configuration](#).

Saving the current configuration

You can save the current configuration by using either of the following:

- [Version 4.0 migration tools](#). This technique exports the existing repository and saves the necessary files. This method is recommended over the alternative, because logic is provided to save the correct files in the structure required by the other migration tools.
- The [Version 3.x XMLConfig tool](#).

3.2.2.2.1: Saving the current configuration by using migration tools

A set of migration tools is provided with the product to help migrate system configurations for installations at Version 3.x and later. The product installation program calls these tools as part of automated migration support. You can call them yourself from the command line.

The tool that saves the system configuration is called *WASPreUpgrade*. This tool saves all files from the following directories in the existing Version 3.x configuration to a specified backup directory:

- hosts
- servlets
- classes
- deployableEJBs
- deployedEJBs
- properties

WASPreUpgrade also saves selected files from the Version 3.x bin directory. Later, the backup directory can be used with the *WASPostUpgrade* tool to restore the previously saved environment into a Version 4.0 installation.

WASPreUpgrade also exports the existing application server configuration from the repository. If you are migrating from an Advanced Edition installation, this step requires that the administration server of the existing environment be running. This command is not typically run from the command line unless both Version 3.x and Version 4.0 are installed on the same machine at the same time.

WASPreUpgrade parameters

```
com.ibm.websphere.migration.preupgrade.WASPreUpgrade backupDirectoryName
currentWebSphereDirectory administrationNodeName[-nameServiceHost host_name [-nameServicePort
port_number]][-traceString trace_spec [-traceFile file_name]]
```

The first three parameters are required and positional. The others are optional. A summary follows:

- `backupDirectoryName` - The name of the directory in which to store the saved configuration and files. The directory is created if it does not already exist. This is a required parameter.
- `currentWebSphereDirectory` - The name of the installation root directory for the current Version 3.x installation. This can be either a Standard or Advanced Edition installation. This is a required parameter.
- `administrationNodeName` - The name of the administration node for the currently installed product. XMLConfig is called using this parameter. This is a required parameter.
- `[-nameServiceHost host_name [-nameServicePort port_number]]` - If specified, these optional parameters are passed to the XMLConfig tool. They can be used to override the default host name and port number used by XMLConfig.
- `[-traceString trace_spec [-traceFile file_name]]` - These optional parameters are used to gather trace information for use by IBM service personnel. The value of the `traceString` parameter is `"*=all=enabled"` and must be specified with quotation marks to be processed correctly.

Logging

The *WASPreUpgrade* tool displays status to the screen while it is running. It also saves a more extensive set of logging information in the backup directory. This file, `WASPreUpgrade.log`, can be viewed with a text editor.

Special instructions for Linux and Solaris installations

When migrating from Version 3.0x on Linux and Solaris installations, mapping problems will occur in the security settings. The resulting configuration contains substitution variables such as `$server_password$` and `$server_root$`. These values must be modified to their correct values after migration has been completed. It is also advisable to migrate these installations with Security disabled.

Restoring the previous Version 3.x configuration

With one minor modification, you can use the `websphere_3x_backup.xml` file (found in the backup directory) with a Version 3.x XMLConfig tool to restore the previous configuration. The version of XMLConfig used by *WASPreUpgrade* encrypts passwords during export. Before you import the configuration back into the Version 3.x installation, these passwords must be reset to their correct, unencrypted values.

Important: This file cannot be used directly in the Version 4.0 environment, because the XMLConfig tool is available only in Version 4.0 Advanced Edition. In addition, the DTDs for the two versions of XML configuration are not compatible; the XML data files exported by the Version 3.x XMLConfig tool cannot be processed by the Version 4.0 tool.

3.2.2.2.2: Saving the current configuration manually

This process uses a Version 3.x XMLConfig tool to export the current configuration.

Before exporting an Advanced Edition configuration to a file, be sure that the administrative server is running. See XMLConfig documentation for further details on the following parameters:

- adminNodeName
- export
- nameServiceHost
- nameServicePort

Export from Version 3.x by using the standard XMLConfig interface

A sample export command for Version 3.x follows:

```
XMLConfig -export j:\websphere\backup\websphere_3x_backup.xml -adminNodeName cally
```

This export technique works with the following limitations:

- The passwords are not encrypted.

The migration version of XMLConfig encrypts passwords; the standard XMLConfig tool does not. Data import is still handled correctly by the migration tools later, but you should be aware of the security exposure of having passwords in the file unencrypted.

- Security data may not be saved.

Prior to Version 3.02.2, XMLConfig did not support the export of security information.

- Substitution keys may be used.

The migration version of XMLConfig encrypts user passwords instead of substitution strings for passwords. This can be resolved by using the `-substitute` keyword when using the WASPostUpgrade tool later.

Export using the migration version of XMLConfig

Important: This procedure does not work for Version 3.x installations on Linux or Solaris. Use the previous procedure (standard Version 3.x XMLConfig).

A sample export command for versions 3.02.x through 3.5.1 follows. For versions 3.5.2 and later, use the XMLConfig35.jar file instead of XMLConfig302.jar.

This sample reflects a Windows NT installation. You may have to update many of the values used in this sample to reflect your configuration requirements or when using a different operating system. The `server.root` variable points to the currently installed Version 3.02x directory.

```
j:\jdk1.1.8.orig\bin\java -Dserver.root=j:\websphere\appserver302-Dcom.ibm.CORBAConfigURL=file:/j:/WebSphere/AppServer302/properties/sas.client.props-classpath
j:\WebSphere\AppServer40\bin\XMLConfig302.jar; j:\websphere\appserver302\lib\ibmwebas.jar;
j:\websphere\appserver302\lib\servlet.jar;
j:\websphere\appserver302\lib\xml4j.jar; j:\websphere\appserver302\lib\ujc.jar;
j:\websphere\appserver302\lib\ejb.jar; j:\websphere\appserver302\lib\console.jar;
j:\websphere\appserver302\lib\admin.jar; j:\websphere\appserver302\lib\repository.jar;
j:\websphere\appserver302\lib\ssligh.jar; j:\websphere\appserver302\lib\tasks.jar;
j:\jdk1.1.8.orig\lib\classes.zip;
j:\websphere\appserver302\propertiescom.ibm.websphere.xmlconfig.XMLConfig-adminNodeName
cally-nameServiceHost cally-nameServicePort 900-export j:\websphere\backup\websphere_3x_backup.xml
```

Backing up configuration files

First, make copies of key directories. Remember that you must update many of the names shown in the following samples to reflect your configuration requirements:

```
j:\websphere\appserver\hosts >
j:\websphere\backup\websphere_3x_backup.userFiles\hosts j:\websphere\appserver\servlets >
j:\websphere\backup\websphere_3x_backup.userFiles\servlets j:\websphere\appserver\classes >
j:\websphere\backup\websphere_3x_backup.userFiles\classes j:\websphere\appserver\deployableEJBs >
j:\websphere\backup\websphere_3x_backup.userFiles\deployableEJBs j:\websphere\appserver\deployedEJBs
> j:\websphere\backup\websphere_3x_backup.userFiles\deployedEJBs j:\websphere\appserver\properties
> j:\websphere\backup\websphere_3x_backup.programFiles\properties
```

Next, back up the following file:

```
j:\websphere\appserver\bin\admin.config >
j:\websphere\backup\websphere_3x_backup.programFiles\bin
```

3.2.2.2.3: Restoring the previous configuration to the new installation

A set of migration tools is provided with the product to help migrate system configurations for installations at Version 3.x and later. The product installation program calls these tools as part of automated migration support. You can call them yourself from the command line.

The tool that restores the Version 3.x configuration is called *WASPostUpgrade*. This tool uses the information created by the *WASPreUpgrade* tool to restore the previous Version 3.x configuration to a Version 4.0 installation.

Because the Version 4.0 product adheres to the J2EE programming model and earlier versions do not, significant changes are required to apply the Version 3.x configuration to a Version 4.0 installation.

Creating and deploying J2EE applications

The J2EE programming model specifies an architecture for how applications are created and deployed. Because applications in Version 3.x were not architected in the same manner, the migration process re-creates these applications. All migrated Web resources and enterprise beans are created in J2EE applications. All enterprise applications defined in the Version 3.x installation are mapped into J2EE applications with the same name and deployed in the default server. All other Web resources and enterprise beans that are mapped but not included in an enterprise application are mapped into a default J2EE application called *DefaultApplication*.

Web applications are mapped to J2EE WAR files. Enterprise beans are deployed as EJB 1.1 beans in J2EE JAR files. These resources are combined in a J2EE EAR file and deployed in the Version 4.0 configuration. There are some differences between the EJB 1.0 and EJB 1.1 specifications, but in most cases, EJB 1.0 beans can run successfully as EJB 1.1 beans. It is recommended that you carefully analyze *WASPostUpgrade.log* (see the end of this topic), because detailed information specific to each bean deployed is saved in the log.

Security

Security settings that were applicable in the Version 3.x environment are applied to J2EE security attributes as part of the migration process.

Samples

Samples are not migrated; they have been updated specifically for J2EE in Version 4.0. The new samples should be used instead of the ones previously provided with Version 3.x product.

Mapping details

See the Related information for more specific information about how objects and attributes are mapped to the Version 4.0 configuration.

WASPostUpgrade parameters

If the configuration is being restored on an Advanced Edition installation, the administrative server must be running for this command to run successfully.

```
com.ibm.websphere.migration.postupgrade.WASPostUpgrade backupDirectoryName [-import
xml_data_file] [-adminNodeName primary_node_name] [-nameServiceHost host_name [ -nameServicePort
port_number]] [-traceString trace_spec [-traceFile file_name]] [-substitute
"key1=value1[:key2=value2;[...]]"]
```

The first parameter of the command is required. The others are optional, except as noted. A summary follows:

- `backupDirectoryName` - The name of the directory that contains the saved configuration and files created by the *WASPreUpgrade* command. This is a required parameter.
- `[-import xml_data_file]` - This optional parameter can be used to specify an XML data file that was created by using the Version 3.x XMLConfig tool. If this parameter is not specified, the default XML configuration file (`websphere_3x_backup.xml`) in the backup directory is used.
- `[-adminNodeName primary_node_name]` - The name of the administrative node for the current installation. This parameter is required if the configuration is being restored on an Advanced Edition installation. XMLConfig is called with this parameter.
- `[-nameServiceHost host_name [-nameServicePort port_number]]` - If specified, these optional parameters are passed to the XMLConfig call. They can be used to override the default host name and port number used by XMLConfig. These parameters are used only if the configuration is being restored on an Advanced Edition installation.
- `[-traceString trace_spec [-traceFile file_name]]` - These optional parameters are used to gather trace information for use by IBM service personnel. The value of `traceString` is `"* =all=enabled"` and must be specified with quotation marks to be processed correctly.
- `[-substitute "key1=value1[:key2=value2;[...]]"]` - If specified, this optional parameter is passed to the XMLConfig call. It is used for

substitution of security values in the XML data file. In the input XML file, each key should appear as \$key\$ for substitution.

Logging

The WASPostUpgrade tool displays status to the screen while it is running. WASPostUpgrade also saves a more extensive set of logging information in the logs directory. The file, WASPostUpgrade.log, can be viewed with a text editor.

3.2.2.2.4: Mapping configurations to Version 4.0

This section details how objects and attributes are mapped to the Version 4.0 environment when you restore a configuration from an earlier product version.

- Directories stdin, stdout, and stderr; passivation directory and working directory

Because the typical location for these directories might include Version 3.x installation directories and the location might be different in the new Version 4.0 installation, additional checking is done for these entries if they are specified. Changed from Version 3.x, the default location for stdin, stdout, and stderr is the logs directory in Version 4.0 installations. Existence of the passivation and working directories is checked before the directories are mapped. If they exist, they are used; otherwise, appropriate defaults are used instead.

- Enterprise beans

EJB 1.0 was the only specification level supported in Version 3.x; EJB 1.1 is the only level supported in Version 4.0. However, many EJB 1.0 beans can be deployed as EJB 1.1 beans successfully. Enterprise beans are redeployed automatically as part of the application migration phase. Be sure to check WASPostUpgrade.log for details of the deployment of these beans; make required changes and redeploy.

- JDBC providers and datasources

JDBC and DataSource objects are significantly redefined in Version 4.0. These objects are mapped to the new configuration by using the Version 3.x settings as input variables.

You might notice a difference between the datasources mapped from Version 3.x and those defined by the samples. The difference is in the user ID and password fields of the datasource. The samples provide a default user ID and password, but the migrated datasources do not. This is because user ID and password data is defined in enterprise-bean bindings, not in the datasource. In Version 3.x, the information is defined at the container and EJB level and so must be mapped to the enterprise bean.

- JSP levels

JSP 0.91 is not supported in Version 4.0. JSP objects that are configured to run as JSP 0.91 are not migrated, but they are noted in the output and logged. JSP 1.0 and 1.1 objects are run as JSP 1.1, because that is the only supported JSP level in Version 4.0.

- Models and clones

Models and clones have been dramatically redefined in Version 4.0. Application servers are the only objects supported as models and clones in Version 4.0. This is a significant difference from Version 3.x, in which many objects could be models and clones. All models and clones relating to application servers are mapped to server groups in Version 4.0.

During the migration of all other objects that were previously clonable, special mapping occurs. All clones are treated as simple objects and are mapped as if they were not clones. Models that are not application server models are ignored and not mapped.

- Multiple application servers

In Version 4.0 Advanced Single Server and Advanced Developer editions, only one application server is configured at one time. In Version 3.x, there can be many application servers defined at one time. During migration of these objects to one of these Version 4.0 editions, the names of the application servers determine how migration occurs. If the names of the application servers match (for example, Default Server), the attributes of the Version 4.0 object are updated to match the previous configuration, and all children are migrated into that application server. If the names do not match, just the children of that Version 3.x application server are migrated to the one application server in the Version 4.0 environment.

- Node name

A Version 3.x repository can contain more than one node name and its associated children. The WASPostUpgrade tool processes only those objects and children that match the node that is being migrated. This determination is made by checking the names of nodes in the configuration files with fully qualified and nonqualified network names of the machine that is being migrated.

- Servlet Redirector

Servlet Redirector is not supported in Version 4.0; these objects are ignored.

- Transports

The default transport type of the Servlet Engine in Version 3.x was Open Servlet Engine (OSE). Because OSE transport is no longer supported in Version 4.0, these transports are mapped to HTTP transports using the same port assignments.

- datasources.xml

In Version 3.x, a datasources.xml file could be used to augment datasource configuration settings. This file was stored in the properties directory. If this file exists, the properties in the file are merged into the configuration of the datasource and JDBC provider configuration.

Other mapping information

In a Version 3.02.x environment, the arguments field in the Default Server object might contain a value of -mx128m. This value is not used in the Version 4.0 environment and is ignored. This value can be removed from the application server arguments field.

In Version 3.02.x, the name of the web/examples/showCfg servlet class was ServletEngineConfigDumper. In versions 3.5 and later, the class name is com.ibm.websphere.examples.ServletEngineConfigDumper. Be aware of this name change if you plan to use this class in the Version 4.0 environment.

3.2.3: Migrating from Version 4.0 Advanced Single Server Edition

A summary of the migration process follows. Most of this is done for you by the product installation program.

1. [Install the Version 4.0 Advanced Edition product.](#)
2. If automated migration is not chosen in the previous step, manually migrate the single-server configuration to the new installation.
3. Restore user data files in the new installation.
4. Uninstall the single-server product.

The rest of this article primarily addresses step 2.

Important: If your current installation uses a configuration file other than `server-cfg.xml`, you must manually migrate the configuration as described in the next section.

Migrating administrative configurations manually

A tool for migrating Version 4.0 Advanced Single Server Edition administrative configurations is provided. There are two ways to migrate these configurations:

- Use the automated migration that is part of product installation. For details, see the related information.
- Manually complete the same steps as the automated migration support would. This might be necessary for nonstandard installations. Information to support this task follows.

The manual migration process

This process uses the `SEMigrator.bat` file (`SEMigrator.sh` on UNIX platforms). This file can be found in the `bin` directory of the Advanced Edition installation.

If the single-server installation uses Default Server for its applications, the default configuration must be installed in the Advanced Edition installation. The `SEMigrator` tool does not migrate default configurations, and the absence of Default Server in the Advanced Edition configuration might cause migration to fail.

Do not uninstall the single-server installation: Preserve it as it is until migration is finished.

1. Run the `SEMigrator` tool from the `bin` directory of the Advanced Edition installation. As discussed in the following section, either the tool creates an output XML file for use in the following step, or the tool completes a one-step migration that is comparable to a full automated migration.
2. If the previous step creates an output XML file (for example, `output.xml`), run the `XMLConfig` tool to import the configuration, as follows:

```
XMLConfig -import output.xml -adminNodeName admin_node_name
```
3. Copy the relevant application EAR directories from the `installedApps` directory of the single-server installation to the Advanced Edition `installedApps` directory. Be sure not to copy EAR directories for the sample applications; overwriting the Advanced Edition sample application configuration will probably result in erratic behavior.

This completes the configuration migration process. You might see certain errors or warnings at the command window if the `SEMigrator` tool tries to re-create configuration objects that already exist in the Advanced Edition installation (for example, JDBC providers or enterprise applications).

The `SEMigrator` tool does not migrate sample applications from the single-server installation. If installation of sample applications and the default configuration is required, please choose the appropriate options during Advanced Edition installation.

SEMigrator parameters

In the following section, *AE* refers to Advanced Single Server Edition. *AE* refers to Advanced Edition.

```
SEMigrator -xmiFile xml_data_file -adminNodeName primary_node_name [ -srcNodeName  
AEs_node_name ] [ -destNodeName AE_node_name ] [ -outputFile output_XML_file_name ] [  
-nameServiceHost host_name [ -nameServicePort port_number ] ] [-traceString trace_spec [-traceFile  
file_name]]
```

A summary of these parameters follows:

- `-xmiFile xml_data_file` - Fully qualified path name for the Advanced Single Server Edition configuration XML file, typically `config\server-cfg.xml`.
- `-adminNodeName primary_node_name` - The name of the node on which the Advanced Edition administrative server is installed. The administrative server must be running if the `outputFile` parameter is not specified.
- `[-srcNodeName AEs_node_name]` - The source node name for migration, that is, the name of a node in `xml_data_file`. This defaults to the first node in `xml_data_file`.

- [-destNodeName AE_node_name] -The destination node name for the migration. This defaults to primary_node_name.
- [-outputFile output_XML_file_name] -The use of this option makes migration a two-step process.

If this parameter is specified, the SEMigrator tool creates an XML file with the specified name, which can then be imported by using XMLConfig to finish the migration. For more information on XMLConfig, see the related information.

The absence of this parameter makes this process equivalent to an automated migration; it requires a running administrative server on the primary node.

- [-nameServiceHost host_name [-nameServicePort port_number]] -The host and port address for the naming server. This pair must be specified if one-step migration is being attempted and the administrative server is running on a remote node. Unless changed previously by administrative actions, typical settings for these parameters are the host name of the administrative server and 900, respectively.
- [-traceString trace_spec [-traceFile file_name]] -The trace specification for this tool.

3.2.4: Migrating Web server plug-ins

If the WebSphere Application Server Version 3.x plug-in for your web server uses Open Servlet Engine (OSE) transport, you must switch to HTTP transport when migrating to WebSphere Application Server Version 4.0.

There are two ways to migrate from OSE transport to HTTP transport. The following instructions are specific to the Web server being supported and assume that you can successfully migrate existing Web applications:

- When migrating all machines at once
- [When migrating one machine at a time](#)

When migrating all machines at once

All-at-once plug-in migration is the easier of the two methods.

1. Make sure that the Web server installation is at the supported level. If you are unsure what level of Web server product is supported, see the Related information.
2. Make sure that you have the correct plug-ins installed on the Web server machine.
 - If the Web server and application server are installed on the same machine and you selected the appropriate Web server during application server installation, further updates are probably not necessary.
 - Otherwise, run the application server installation program on the Web server machine and select **Custom** installation. Then select only the Web server plug-ins for installation. This installs the plug-ins needed to run and makes the necessary configuration changes for the supported Web servers.
3. Migrate the machines from Version 3.x to Version 4.0.
4. Regenerate the plug-in configuration file.
 - If the Web server and application server are running on the same machine, further changes to the plug-in configuration file are probably not necessary.
 - Otherwise, make sure that the hostname attribute for the transports is set to the host name or IP address on which the application server is running.
5. Move the plugin-cfg.xml file to the correct location in the Web server installation.

When migrating one machine at a time

It is possible to run the Version 3.x OSE plug-in and the Version 4.0 HTTP plug-in in the same Web server installation. This enables you to keep a cluster of Version 3.x machines and a cluster of Version 4.0 machines as you migrate machines one at a time from Version 3.x to Version 4.0. The instructions for incremental migration vary by Web server product and by platform. In all cases, however, the URI for an machine must be unique in the routing rules for the plug-in. For example, you can't have `/servlet/*` defined in both the OSE properties file and the HTTP configuration file. If there is duplication, operating behavior will be erratic.

On Windows, make sure that the directory that contains the `plugin_common.dll` file has been added to the system path. (Otherwise, the Web server plug-ins will not load.) It might be necessary to reboot the computer after you have updated this environment variable.

Plug-in migration has been tested with the following Web server products:

- IBM HTTP Server
- [iPlanet](#)

- Lotus Domino
- Microsoft IIS

Migration for IBM HTTP Server

1. Move the appropriate files from the bin directory of a Version 4.0 application server installation.
 - In Windows, move mod_ibm_app_server_http.dll and plugin_common.dll
 - In Unix, move mod_ibm_app_server_http.so(sl)
2. To the httpd.conf file, add the lines for configuring the Version 4.0 Web server plug-in (/http/webservers.html).
3. Regenerate the plug-in configuration file on the application server machine after you have the machine migrated. Be sure the hostname attributes of the transports are set to the host name or IP address of the machine on which the application server is running.
4. Move the plug-in configuration file into the Web server installation so that it is in the location specified by the WebSpherePluginConfig directive in the httpd.conf file.
5. Restart the Web server; you should be able to access applications that run on both Version 3.x and Version 4.0 application server clusters.
6. As you migrate more machines over to Version 4.0, you must regenerate the Web server plug-in configuration after migration and move the plug-in configuration file to the Web server installation. You should also remove the machine from the OSE files by either manually editing them or removing the machine from the Version 3.x instances and then regenerating the OSE property files.

Migration for iPlanet

1. Move the appropriate files from the bin directory of a Version 4.0 application server installation.
 - In Windows, move ns41_http.dll and plugin_common.dll
 - In Unix, move libns41_http.so(sl)
2. To the obj.conf file, add the lines for configuring the Version 4.0 Web server plug-in (/http/webservers.html).
3. Regenerate the plug-in configuration file on the application server machine after you have the machine migrated. Be sure the hostname attributes of the transports are set to the host name or IP address of the machine on which the application server is running.
4. Move the plug-in configuration file into the Web server installation so that it is in the location specified by the bootstrap.properties variable for the Init directive in the obj.conf file.
5. Restart the Web server; you should be able to access applications that run on both Version 3.x and Version 4.0 application server clusters.
6. As you migrate more machines over to Version 4.0, you must regenerate the Web server plug-in configuration after migration and move the plug-in configuration file to the Web server installation. You should also remove the machine from the OSE files by either manually editing them or removing the machine from the Version 3.x instances and then regenerating the OSE property files.

Migration for Lotus Domino

Make sure that the registry is set correctly. If you have not previously installed the Version 4.0 application server on the Web server machine, add a key called **4.0** to **WebSphere Application Server** and then add the Plugin Config variable to the key for the Web server plug-in to load. The complete instructions for this can be found in the instructions for manually configuring the Web server plug-in (/http/webservers.html).

1. Move the appropriate files from the bin directory of a Version 4.0 application server installation.
 - In Windows, move domino5_http.dll and plugin_common.dll
 - In Unix, move libdomino5_http.a(so,sl)
2. Follow the steps for manually configuring the Version 4.0 Web server plug-in (/http/webservers.html).
3. Regenerate the plug-in configuration file on the application server machine after you have the machine migrated. Be sure the hostname attributes of the transports are set to the host name or IP address of the machine on which the application server is running.
4. Move the plug-in configuration file into the Web server installation so that it is in the location specified by the Plugin Config variable that you added to the registry.
5. Restart the Web server; you should be able to access applications that run on both Version 3.x and Version 4.0 application server clusters.
6. As you migrate more machines over to Version 4.0, you must regenerate the Web server plug-in configuration after migration and move the plug-in configuration file to the Web server installation. You should also remove the machine from the OSE files by either manually editing them or removing the machine from the Version 3.x instances and then regenerating the OSE property files.

Migration for Microsoft IIS

Make sure that the registry is set correctly. If you have not previously installed the Version 4.0 application server on the Web server machine, add a key called **4.0** to **WebSphere Application Server** and then add the Plugin Config variable to the key for the Web server plug-in to load. The complete instructions for this can be found in the instructions for manually configuring the Web server plug-in (/http/webservers.html).

1. Move the files iisWASPlugin_http.dll and plugin_common.dll from the bin directory of a Version 4.0 application server installation.
2. Follow the steps for manually configuring the Version 4.0 Web server plug-in (/http/webservers.html).
3. Regenerate the plug-in configuration file on the application server machine after you have the machine migrated. Be sure the hostname attributes of the transports are set to the host name or IP address of the machine on which the application server is running.
4. Move the plug-in configuration file into the Web server installation so that it is in the location specified by the Plugin Config variable that you added to the registry.
5. Restart the Web server; you should be able to access applications that run on both Version 3.x and Version 4.0 application server clusters.
6. As you migrate more machines over to Version 4.0, you must regenerate the Web server plug-in configuration after migration and move the plug-in configuration file to the Web server installation. You should also remove the machine from the OSE files by either manually editing them or removing the machine from the Version 3.x instances and then regenerating the OSE property files.

3.3: Migrating APIs and specifications

IBM WebSphere Application Server supports a wide variety of technologies for building powerful enterprise applications. As technology advances, particularly in the area of Java components, new Application Server product versions advance to support and extend the most contemporary open specification levels.

If your existing applications currently support different specification levels than are supported by this version of the product, it is likely you will need to update at least a few aspects of the applications to comply with the new specifications.

In many cases, IBM extends the specification levels that are currently supported by the product to provide additional features and customization options. If your existing applications use extensions from earlier product versions, mandatory or optional migration could be necessary to utilize the same kinds of extensions in the current version.

From Version 3.0x to Version 4.0, the main migration areas concern the IBM extensions and the JDK. In contrast, migrating from Version 2.0x requires updating applications with respect to the open specifications, such as the Java Servlet API.

The table summarizes potential migration areas. See the Related information below for instructions pertaining to each area.

Functional area	Support in current version	Need to migrate from V3.x?	Need to migrate from V2.0x?	Details
Enterprise beans	EJB 1.0 Specification	Yes*	Yes**	* Many EJB 1.0 applications will run unchanged in Version 4.0; however, some changes may be required or recommended. See article 3.3.1 for more information. ** Version 2.0x offered only limited EJB 1.0 Specification support
Servlets	Servlet 2.1 Specification and IBM extensions	Yes*	Yes	Article 4.2.1.2.1a describes the Servlet 2.2 APIs. * Many Servlet 2.1 applications will run unchanged in Version 4.0; however, some changes may be required or recommended.
Servlets	Servlet 2.2 Specification	No	not applicable	Article 4.2.1.2.1a describes the new Servlet 2.2 APIs. Version 2.0x supported the Servlet 2.0 Specification.
JSP files	JSP .91 Specification	Yes	Yes	JSP 0.91 is not supported in Version 4.0.

JSP files	JSP 1.0 Specification	Yes*	Yes	<p>* Many JSP 1.0 applications will run unchanged in Version 4.0; however, some changes may be required or recommended.</p> <p>Version 2.0x only supported the JSP .91 Specification.</p>
JSP files	JSP 1.1 Specification	No	not applicable	Version 2.0x only supported the JSP .91 Specification.
XML	XML 2.0.x supported	Yes	Yes	See article 3.3.4 for migration requirements.
JDBC and IBM database connection support APIs	JDBC 2.0; connection pooling model	Yes	Yes	<p>V2.0x supported JDBC 1.0 and a connection manager model.</p> <p>If still using Connection Manager, you must switch to connection pooling.</p> <p>Do not forget to switch to supported JDBC 2.0 drivers.</p>
User profiles	IBM user profile APIs	No	Yes	Need to migrate from V2.0x deprecated classes for use with V4.0.
Sessions	IBM session support APIs	No	Yes	Need to migrate from V2.0x deprecated classes, changes to clustering, URL encoding for use with V4.0.
Security	IBM security support	No	No	<p>No action required. However, unlike previous versions, Version 4.0 does not protect URIs served by an external Web server. One way of protecting these is by using the WebSeal product; see article 5.6.</p> <p>Version 4.0 continues to protect URIs (including URIs for HTML files) that are served by the application server.</p>

Transactions	Java 1.2 transactions support	Yes	Yes	Version 3.0x provided proprietary IBM packages to simulate Java 1.2 functionality. Version 2.0x did not provide any support. Migrate to Version 4.0 if your applications require this kind of support.
XML configuration	XMLConfig tool	Yes	Yes	The XML Configuration Management Tool (XMLConfig) was introduced in Version 3.02. The DTD and many of the interfaces have changed in Version 4.0. XMLConfig is not supported in Version 4.0 Advanced Single Server Edition.
WebSphere Control Program	WSCP	Yes	not applicable	The WebSphere Control Program (WSCP) was introduced in Version 3.5. Many of the interfaces have changed in Version 4.0; see online help . WSCP is not supported in Version 4.0 Advanced Single Server Edition.

3.3.1: Migrating to supported EJB specification

Migrating from Version 3.x

The EJB specification level for Version 4.0 changed from that of Version 3.x. In Version 4.0, the EJB specification level is EJB 1.1, not EJB 1.0. Changes due to the prerequisite of JDK 1.3 are also required:

- In Version 3.02x, the JavaSoft standard packages:

`javax.sql.*` and `javax.transaction.*`

were present under non-standard names. In Version 4.0, they are present under their standard names.

Any code using WebSphere Application Server data sources, including BMP entity beans and session beans that access databases, will need to be modified.

See articles 3.3.8 and 3.3.9 for instructions.

- Some of the stub classes for deployed enterprise beans have changed in JDK 1.3. [Redeploy all EJB server JAR files](#) to generate the correct stub file references.

Be aware that, in general, JAR files generated prior to JDK 1.3 are source and binary code compatible on a JDK 1.3 base. However, there are some incompatible cases. For further details, see:

<http://java.sun.com/products/jdk/1.3/compatibility.html>

Important: The task of deploying enterprise beans has changed significantly. Start at [article 6.3](#) for an overview of the assembly and deployment process.

3.3.2: Migrating to supported Servlet specification and extensions

Servlets will require migration if they are not of the supported specification level (2.2) or they rely on deprecated or removed IBM servlet extensions.

Migrating to the supported Servlet specification

Refer to the [Java Servlet API 2.2 specification](#) for complete information concerning new and deprecated APIs. [Article 4.2.1.2.1a](#) highlights a few of the new and deprecated classes and methods.

3.3.2.1: Example: Migrating `HttpServletResponse.callPage()`

Calls to `HttpServletResponse.callPage()` need to be replaced by calls to `RequestDispatcher`, as shown.

Before -- Using `HttpServletResponse.callPage()`

```
import java.io.*;import javax.servlet.*;import javax.servlet.http.*;public class UpdateJSPTest
extends HttpServlet{    public void doGet (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException        {            String message = "This is a test";
((com.sun.server.http.HttpServletRequest)req).setAttribute("message", message);
((com.sun.server.http.HttpServletResponse)res).callPage("/Update.jsp", req);        }}
```

After -- Using `RequestDispatcher`

```
import java.io.*;import javax.servlet.*;import javax.servlet.http.*;public class UpdateJSPTest
extends HttpServlet{    public void doGet (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException        {            String message = "This is a test";
req.setAttribute("message", message);            RequestDispatcher rd =
getContext().getRequestDispatcher("/Update.jsp");            rd.forward(req, res);
//((com.sun.server.http.HttpServletRequest)req).setAttribute("message", message);
//((com.sun.server.http.HttpServletResponse)res).callPage("/Update.jsp", req);        }}
```

3.3.4: Migrating to supported XML API

If your XML applications use XML for Java API Version 2.0.x or earlier, you must migrate them to API Version 3.1 or the equivalent open-source version.

Although there are inherent performance improvements in later versions of the XML for Java API, you can gain additional performance by explicitly using nonvalidating parsers in application environments where the data can be trusted.

Issues for migrating from XML for Java API Version 1.1.x

The most significant change is that the TX-compatible APIs are no longer available. The Document API retains the XML manipulation APIs that were in TXDocument, but the following functionality must be rewritten:

- Creating and loading an XML parser: We suggest the use of a Java API for XML Processing (JAXP) factory class.
- Writing out the DOM tree: A serializer must now be used.

One drawback to the DOM Level 2 implementation in this level of the XML for Java API is that the grammar (DTD or schema) is no longer a node in the DOM tree, so it cannot be written out. As a result, we recommend that only external grammars be used. You can query the system ID of the root element and use it to retrieve the name that is specified in the `<!DOCTYPE>` statement. After the tree has been written out to an XML file, you can read the file as text and insert a `<!DOCTYPE>` statement.

3.3.5: Migrating to supported user profile APIs

Migrating from Version 3.x

Changes to code are not required. However, configuration of the user profile has changed in Version 4.0. See the Related information for instructions.

Migrating from Version 2.0x

The user profile implementation in versions 3.x and later differs significantly from that in Version 2.0, as follows:

Profile management functions

The user profile management functions are separated from the data elements (the elements mapped to the columns in the database schema).

The management functions in the `com.ibm.websphere.userprofile.UserProfile` class are deprecated and disabled. The class is to be used solely for getting and setting data for individual instances of users.

Extending the base implementation

You can now extend the base user profile implementation to include custom database columns and import legacy databases. See the Related information for instructions.

3.3.6: Migrating session management

Migration from Version 3.x

Some API changes are required in order to redeploy existing applications on Version 4.0. These include changes to the HttpSession API itself as well as issues associated with moving to support for the Servlet 2.2 specification.

Certain access methods have been deprecated in Version 4.0. These deprecated APIs still work in Version 4.0, but they may be removed in a future version of the product. Changes are summarized in the following table:

Wherever you use...	Use this instead
getValue()	getAttribute()
getValueNames()	getAttributeNames()
removeValue()	removeAttribute()
putValue()	setAttribute()

In accordance with the Servlet 2.2 specification, HttpSession objects must be scoped within a single Web application context; they may not be shared between contexts. This means that a session can no longer span Web applications. Objects added to a session by a servlet or JSP in one Web application cannot be accessed from another Web application. The same session ID may be shared (because the same cookie is in use), but each Web application will have a unique session associated with the session ID.

Relative to session security, the default Session Manager setting for Integrate Security is now *false*. This is different from the default setting in some earlier releases.

In addition, you may want to review the tuning options now available ([article 4.4.1.1.7](#)).

In previous releases of the product, JSPs that contained the usebean tag with scope set to *session* did not always work properly when session persistence was enabled. Specifically, the JSP writer needed to write a scriptlet to explicitly set the attribute (that is, call `setAttribute()`) if it was changed as part of JSP processing. Two new features in Version 4.0 help address this problem:

- You can set `dosetAttribute` to *true* on the [JSP InitParameter](#).
- You can set the [Write Contents](#) option to *Write all*.

The differences between the two solutions are summarized in the following table:

	Applies to...	Configured at ...	Action
<code>dosetAttribute</code> set to <i>true</i>	JSP	JSP enabler	Assures that JSP session-scoped beans always call <code>setAttribute()</code>
Write Contents option set to <i>Write all</i>	servlet or JSP	application server	All session data (changed or unchanged) is written to the database

If session persistence is enabled and a class reload for the Web application occurs, the sessions associated with the Web application are maintained in the persistent store and will be available after the reload.

Some of these differences might impact how you choose to track and manage sessions.

Migration from Version 2.0x

Relative to Version 2.0x, Version 3.0x introduced some changes to session support. See the [Related information](#).

3.3.6.1: Migrating from Version 2.0 session support

Note these changes to the implementation of sessions in IBM WebSphereApplication Server Version 2.x. These changes are in addition to those listed in article 3.3.6.

- The public classes in the `com.ibm.servlet.personalization.sessiontracking` package have been deprecated.

Application developers can still compile servlets using the old classes. (Specifically, the `IBMSessionData` class typecast still works). However, the functions will return null or constant values, and no processing or setting of values will occur.

- Clustering is now handled using a database or session affinity, and the Version 2.0 concept of session cluster client and server is no longer valid because all nodes within a cluster are now considered equal.
- Extensions for sessions to the Java Servlet API are now encapsulated in the `com.ibm.websphere.servlet.session.IBMSession` interface.
- If URL encoding is configured and `response.encodeURL()` or `encodeRedirectURL()` is called, the URL is encoded, even if the browser making the HTTP request processes cookies. This differs from the behavior in previous releases, which checked for the condition and halted URL encoding in such a case.

3.3.7: Migrating to supported security APIs

No action is required.

3.3.8: Migrating to supported database connection APIs (and JDBC)

Connection pooling (provided through DataSource objects) was introduced in IBM WebSphere Application Server Version 3.0x. Applications that use Version 3.0x connection pooling need to be changed slightly and recompiled.

If existing applications are still using the connection manager model from Version 2.0x, you must update the application code to use the current connection pooling model (see the Related information). The shift in models corresponds to a change in supported JDBC specification levels.

ConnectionPreemptedException, introduced in Version 3.0x, no longer exists. StaleConnectionException has replaced ConnectionPreemptedException in all cases. For details, see the Related information.

Items newly deprecated in Version 4.0

The packages `com.ibm.db2.jdbc.app.stext.javax.sql` and `com.ibm.ejs.dbm.jdbcext` have been deprecated. Applications using the `com.ibm.ejs.dbm.jdbcext` package will still be allowed to retrieve a DataSource, but new DataSources cannot be created or bound into JNDI by using this interface. All new DataSources must be created by using `com.ibm.websphere.advanced.cm.factory.DataSourceFactory`.

The following methods in `com.ibm.websphere.advanced.cm.factory.DataSourceFactory` have been deprecated: `createJTADatasource()` and `createJDBCDataSource()`. These methods have been replaced with the `getDataSource(java.util.Properties)` method.

The class `com.ibm.ejs.cm.portability.StaleConnectionException` has been deprecated. Applications currently using this class will still function, but it is recommended that new applications be written using `com.ibm.websphere.ce.cm.StaleConnectionException`.

3.3.8.1: Migrating from the Version 3.0x connection pooling model

Connection pooling (provided through DataSource objects) was introduced in IBM WebSphere Application Server Version 3.0x. Application components that use Version 3.0x connection pooling need to be changed slightly and recompiled. First, replace the following import statement:

```
import com.ibm.db2.jdbc.app.stdebt.javax.sql.*;
```

with this:

```
import javax.sql.*;
```

Connection pooling behavior in versions 3.5 and later changed relative to that in Version 3.0x. If your application typically requires two or more connections to the same database manager, consider the multiple-connection scenarios in [Article 0.14.2](#).

3.3.8.2: Migrating servlets from the connection manager model

Servlets written to use the connection manager must be modified to use WebSphere connection pooling.

For most servlets, the migration consists of simple code changes. Because new servlets cannot use the connection manager, the details of connection manager coding are not discussed, except as needed in the migration.

Migration involves the following activities. For more details, see the related information.

Action needed	From something like ...	To something like ...
Update import statements	<pre>import java.sql.*; import com.ibm.servlet.connmgr.*;</pre>	<pre>import javax.sql.*; import javax.naming.*;</pre>
Modify servlet init() methods	<pre>IBMConnSpec spec = new IBMJdbcConnSpec("poolname", true, "COM.ibm.db2.jdbc.app.DB2Driver", "jdbc:subprotocol:database", "userid", "password"); IBMConnMgr connMgr = IBMConnMgrUtil.getIBMConnMgr();</pre>	<pre>Hashtable parms = new Hashtable(); parms.put(Context.INITIAL_CONTEXT_FACTORY, "com.ibm.websphere.naming.WsnInitialContextFactory"); Context ctx = new InitialContext(parms); DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/SampleDB");</pre> <p>where <i>SampleDB</i> is the name of the data source.</p> <p>The WebSphere administrator provides information on the arguments for the put() and lookup() methods.</p>
Modifying how servlets obtain and close connections	<pre>IBMJdbcConn cmConn = (IBMJdbcConn)connMgr.getIBMConnection(spec); Connection conn = cmConn.getJdbcConnection(); ...cmConn.releaseIBMConnection();</pre>	<pre>Connection conn = ds.getConnection("userid", "password"); ...conn.close();</pre>
Modify preemption handling	Call verifyIBMConnection()	Catch com.ibm.websphere.ce.cm.StaleConnectionException

Considerations for new servlets

The connection manager APIs are not supported in the Application Server Version 4.0 environment. You should not write new servlets using the connection manager. Instead, write new servlets using the current connection pooling model.

3.3.8.3: Obsoleteconnection manager APIs

Some connection manager APIs are intended only for monitoring purposes or internal connection manager use; they do not have any practical use in production servlets. Therefore, such APIs were not migrated to the Application Server Version 3.x environment and are not likely to be found in existing production servlets.

The following table lists the connection manager classes and associated methods that are no longer supported. The classes are now obsolete, so the details of connection manager coding are not discussed.

Obsoleteconnection manager class	Methods
<p>com.ibm.servlet.connmgr.IBMConnMgrUtil</p>	<ul style="list-style-type: none"> ● public static IBMConnMgr getIBMConnMgr() ● public static IBMConnPoolSpec getPoolProperties(String poolName) ● public static void addPoolProperties(IBMConnPoolSpec spec) ● public static String urlToPoolName(String url)
<p>com.ibm.servlet.connmgr.IBMConnMgr</p>	<ul style="list-style-type: none"> ● public IBMConnection getIBMConnection(IBMConnSpec connSpec) ● public IBMConnection getIBMConnection(IBMConnSpec connSpec, String ownerClass)
<p>com.ibm.servlet.connmgr.IBMConnection</p>	<ul style="list-style-type: none"> ● public boolean verifyIBMConnection() ● public void removeIBMConnection() ● public void releaseIBMConnection()
<p>com.ibm.servlet.connmgr.IBMJdbcConn</p> <p>This class is derived from the IBMConnection class above and it implements one additional method, as shown.</p>	<ul style="list-style-type: none"> ● public Connection getJdbcConnection()
<p>com.ibm.servlet.connmgr.IBMConnPoolSpec</p> <p>This class and the associated methods are intended for WebSphere Studio use only. Both methods are constructors.</p>	<ul style="list-style-type: none"> ● public IBMConnPoolSpec(String poolName, String poolType, int maxConnections, int minConnections, int connectionTimeout, int maxAge, int maxIdleTime, int reapTime) ● public IBMConnPoolSpec(String poolName, String poolType)
<p>com.ibm.servlet.connmgr.IBMJdbcConnSpec</p> <p>The first three methods are constructors.</p>	<ul style="list-style-type: none"> ● public IBMJdbcConnSpec(String poolName, boolean waitRetry, String dbDriver, String url, String loginUser, String loginPasswd) ● public IBMJdbcConnSpec(String poolName) ● public IBMJdbcConnSpec() ● public void verify()

3.3.9: Migrating to supported transaction support

Version 3.x of the product ran with a 1.1 level of JDK. Version 3.x included packages written by IBM to provide transaction support features usually provided by JDK 1.3. Now that Version 4.0 runs with JDK 1.3, applications should no longer import the proprietary IBM packages, but instead import the open Java 1.3 packages that provide the required functionality.

1. In Java source files, find the import statement:

```
import com.ibm.db2.jdbc.app.jta.javax.transaction.*
```

2. Change the import statement to:

```
import javax.transaction.*
```

3. Recompile the Java files using JDK 1.3.

Other transaction considerations for Version 4.0:

- One database connection cannot be used across multiple user transactions. If an application component obtains a connection to a database, then begins a transaction, the connection is closed automatically when the transaction ends. The connection must be obtained again before beginning another transaction.
- Transactions that began by using UserTransaction now use the isolation level specified when the enterprise bean is deployed.

In Version 3.02, the transaction isolation level defaulted to:

- REPEATABLE_READ for DB2
- SERIALIZABLE for Oracle
- The timeout units for transaction inactivity are in milliseconds.
- If multiple datasource connections are involved in the same transactions, then JTA must be enabled on those datasources. JTA must be enabled for two-phase commit actions.

3.3.10: Migrating to supported XML configuration

The composition of almost all of the existing elements has changed. In some cases, these pre-existing elements have children of a different type. In other cases, these pre-existing elements have a completely new set of child elements. For details, see `xmlconfig.dtd`.

Changes for some elements are summarized in the following table:

Element	Changed syntax
container	No longer exists; attributes of <code><container></code> are now part of <code><application-server></code>
jdbc-driver	Now supports <code><install-info/></code> and <code><uninstall-info/></code> elements (optional)
model	No longer exists; <code><model></code> has been replaced with <code><server-group></code>
session-manager	This configuration is now part of <code><application-server></code>
data-source	<code><data-source></code> is now contained in <code><jdbc-driver></code> and is no longer a direct child of <code><websphere-sa-config></code>

Changes from Version 3.x

For programmatic access:

- The XMLConfig constructor now throws `NamingException` and `InvalidArgumentException`.
- The XMLConfig tools now supports variable substitution and variable Hashtable setter.

The XMLConfig command line now supports the `-substitute` option for variable replacement.

3.5: Switching administrative databases

Suppose you want to switch from one supported database brand to another, for use as the WebSphere administrative database. Originally, the WebSphere Application Server installation program prompted you to enter information about the database configuration. This article provides instructions for switching to a different database brand without needing to run the WebSphere Application Server installation program again.

Basically, you need to preserve the administrative configurations as stored in the current database, switch the administrative server settings to point to the new database, then import the administrative configurations into the new database. Follow these instructions:

1. Stop the WebSphere administrative server (or servers).
2. Use the `-export` option of the [XMLConfig command line administrative client](#) to export the configuration information from the current administrative database.
3. Assuming you have already installed the database brand to which you are switching, start the database management system for the database. Use the administrative facilities to create an appropriately named database to use as the WebSphere administrative database.
4. Use the [database conversion assistant](#) to guide you through the necessary setting changes.

The Related information describes the settings affected by the database conversion assistant.

5. To have the new database contain configurations for the default application server and other default resources, follow the instructions in [article 6.6.46](#) for an already installed product.
6. Use the `-import` option of the [XMLConfig command line administrative client](#) to import the configuration information that you exported earlier. This will effectively put the contents of the former administrative database into the new administrative database.
7. Start the administrative server.

Besides checking that the tables and data were correctly imported to the new database, performing the last two steps provides verification that the procedure was successful. If the administrative database settings were not modified successfully, the import would fail with SQL errors related to creating tables and data that already exist. (The import would be attempting to write to the former database, from which it exported the data, instead of to the new database). Similarly, attempts to start the administrative server would fail with SQL-related exceptions.

3.5.1: Using the database conversion assistant to switch administrative databases

The database conversion assistant helps you modify the WebSphere administrative server configuration in order to switch from one administrative database to another. This user-friendly editor leads you through the necessary changes to the database configuration information stored in the [administrative server configuration](#) and setup command line files.

 The assistant helps you convert database *settings*. You must also move database *content* from the old database to the new database. (You can use the XMLConfig administrative client to do so automatically). Use the database migration assistant as part of the overall procedure described in [article 3.5](#).

Obtaining and running the database conversion assistant

1. Create a directory named DBUpgrade. Make sure it is a subdirectory of the *WebSphere* directory in the [product installation root](#).
2. Obtain the database conversion assistant file (a ZIP or JAR) from the among the tools offered on the [Support page of the product Web site](#).
3. Extract the database conversion assistant file contents into the DBUpgrade directory.
4. If using Windows NT, add the following to the PATH environment variable:
product_installation_root\jdk\bin
5. Run the assistant:
 - o **UNIX** dbupgrade.sh
 - o **NT** dbupgrade.bat
6. Follow the instructions in the assistant interface.

Troubleshooting

The tool might fail sometimes due to some extra entries in the CLASSPATH or PATH environment variables. In this case, do the following:

1. Open a command prompt window and set CLASSPATH and PATH as follows (so there are no other entries in the CLASSPATH or PATH):
set CLASSPATH=product_installation_root\jdk\lib\tools.jar set PATH=product_installation_root\jdk\bin
2. Run the tool from the command prompt. Once the window is closed, the CLASSPATH and PATH settings will reset to the system settings.

3.5.2: Settings to change when switching administrative databases

The following steps describe the settings modified by the database conversion assistant. It is highly recommended that you use the database conversion assistant to switch database settings, rather than editing the settings by hand.

Note, the following example steps pertain specifically to switching from an Instant DB database to a DB2 database. Again, they are provided as a reference to what the assistant is doing. Use the assistant to modify the settings.

1. Modify the [administrative server configuration file](#), changing entries as follows:

- From:
`com.ibm.ejs.sm.adminServer.dbDriver=jdbc.idbDriver`
To:
`com.ibm.ejs.sm.adminServer.dbDriver=COM.ibm.db2.jdbc.app.DB2Driver`
- From:
`com.ibm.ejs.sm.adminServer.dbUser=`
To:
`com.ibm.ejs.sm.adminServer.dbUser=db2admin`
where db2admin is the appropriate database user name.
- From:
`com.ibm.ejs.sm.adminServer.dbPassword=`
To:
`com.ibm.ejs.sm.adminServer.dbPassword=db2admin`
where db2admin is the appropriate database password.
- From:
`com.ibm.ejs.sm.adminServer.dbUrl=jdbc:ibd:c:/WebSphere/AppServer/bin/was.prp`
To:
`com.ibm.ejs.sm.adminServer.dbUrl=jdbc:db2:was`
- Modify the entry that points to the database driver:
`com.ibm.ejs.sm.adminserver.classpath`
For example, change from:
`c:/WebSphere/AppServer/lib/ibd.jar`
To:
`c:/SQLLIB/java12/db2java.zip`

2. Modify [product_installation_root/bin/setupCmdLine\(setupCmdLine.sh on UNIX\)](#):

- Change from:
`SET DB2DRIVER=c:\WebSphere\AppServer\lib\ibd.jar`
To:
`SET DB2DRIVER=c:\SQLLIB\java\db2java.zip`
as appropriate for the location of the database driver.
- Set the DB2_HOME entry:
`DB2_HOME=/home/db2inst1`
as appropriate for the installation root of the database product.

3. Modify the [administrative server startup script](#):

- Change from:
`DB_TYPE=InstantDB`
To:
`DB_TYPE=DB2`
- Change from:
`DBUSERID=`
To:
`DBUSERID=db2inst1`
- Change from:
`DB_INSTANCE_HOME=`
To:
`DB_INSTANCE_HOME=/home/db2inst1`

4. Modify [product_installation_root/properties/initial_setup.config](#):

- Change from:
`<config-file>product_installation_root/hosts/default_host/WSsamplesIDB_app/WSsamplesIDB_app_create.xml</config-file>`
To:
`<config-file>product_installation_root/hosts/default_host/WSsamples_app/WSsamples_app_create.xml</config-file>`
- Change from:
`<value>jdbc:ibd</value>`
To:
`<value>jdbc:db2</value>`
- Change from:
`<value>product_installation_root/lib/ibd.jar</value>`
To:
`<value>home/db2inst1/sqllib/java/db2java.zip</value>`

o <value>product_installation_root/bin/myidb.prp</value>

To:

<value>WAS</value>

where WAS is the name of the DB2 database.

3.5.3: Switching server databases to DB2/390

You can use DB2/390 to store the HttpSession and administrative databases. You can also use DB2/390 with enterprise beans.

This article does not outline the entire process of switching. As applicable, this article focuses on two steps: database setup in DB2/390 and administrative settings required in the application server.

If you are using DB2/390 in this capacity for the first time, additional configuration in the DB2 Connect product may be necessary. For more information, see the DB2 Connect documentation.

Setting up the HttpSession database in DB2/390

Session Manager now supports DB2/390 as a persistent datastore for failover support. Set up the database as follows:

1. In DB2/390, create the database table.
2. Create indexes for ID and PROPID.
3. Use an SQL script like the following:

```
CREATE TABLE DATABASE.SESSIONS (ID VARCHAR(64) NOT NULL, PROPID VARCHAR(64)
NOT NULL, APPNAME VARCHAR(64), LISTENERCNT SMALLINT, LASTACCESS DECIMAL(21),
CREATIONTIME DECIMAL(21), MAXINACTIVETIME INTEGER, USERNAME VARCHAR(256),
SMALL VARCHAR(3313) FOR BIT DATA, MEDIUM LONG VARCHAR FOR BIT DATA, LARGE BLOB(2M),
TROW ROWID GENERATED ALWAYS ) IN WAS.CMPTB001; CREATE TYPE 2 UNIQUE INDEX
CMP.CMPND001 ON DATABASE.SESSIONS (ID, PROPID) USING VCAT DSN610; COMMIT;
CREATE LOB TABLESPACE CMPLS001 IN WAS BUFFERPOOL BP32K LOCKSIZE LOB USING
VCAT DSN610 CLOSE NO; CREATE AUX TABLE CMP.CMPLT001 IN WAS.CMPLS001
STORES DATABASE.SESSIONS COLUMN LARGE; COMMIT; CREATE INDEX CMP.CMPLI001 ON
CMP.CMPLT001 USING VCAT DSN610; COMMIT;
```

This statement creates the SESSIONS table in the default database, which on DB2/390 is not recommended. Therefore, you should add an IN DATABASE() clause for your particular production requirements.

- In DB2/390, set the RRULOCK parameter to YES.

This parameter setting ensures an update lock on the table, which is required for proper serialization of session data. You can set this parameter by using option 19 on panel DSNTIPB.

- In Session Manager, specify the database and version.

- a. Under the Persistence tab of Session Manager, assign the datasource that points to the DB2/390 location.
- b. Specify the user ID, database, and version in the userid field, as follows:

```
userid::database$VxWhere :: and $V are separators userid is the userid for connecting
to database database is the database name x is the version of the database (values
can be 6 or 7)
```

Example for DB2 Version 6: user91::u091db92\$V6

- c. Provide the password.
- d. Apply the changes.

Setting up the administrative database in DB2/390

Before you start the setup in DB2/390, find the JCL file named db2390.sql, located in the bin directory. This sample is shipped with WebSphere Application Server, Version 3.5.3 and later.

1. At a minimum, customize the JCL as follows:
 - For *wasdb*, substitute the name of the database in which the tables are to be defined.
 - For *hlq*, substitute the high-level qualifier for the datasets. This is used in the DDL so that DB2 can associate the tablespaces with the linear datasets.
 - For *tgtVolume*, substitute the target volume for dataset allocation.
2. Transfer the customized JCL file to the target OS/390 system.
3. Run the customized JCL.

The JCL stream is broken up into three steps, as follows:

1. DROPDB - The expected return code is 0 or 8. Expect a return code of 8 on the first run of this stream, because the database will not yet exist.
2. DEFTABLE - Expect a return code of 0. This step creates the linear VSAM datasets for the tablespaces, indexes, and BLOB tablespaces used by the repository structure.
3. CRTTABLE - Expect a return code of 0. DDL for the tablespaces, tables, and indexes is run in this step. In this step, several GRANT statements allow unlimited access to the tables by PUBLIC. The appropriate security measures must be implemented as required in your installation.
4. Before the administrative server is started, verify the following property setting in the admin.config file:

```
com.ibm.ejs.sm.adminServer.dbInitialized=nocreate
```

This directs the administrative server not to try to create tables when DB2/390 is being used.

Using DB2/390 with enterprise beans

For session or BMP beans, you can use DB2/390 just as you would use any other version of DB2. No additional database setup or administrative configuration is required.

You can also use DB2/390 to store persistent data for CMP beans, but this support is limited to CMP beans developed in IBM VisualAge for Java, Version 3.5.3 or later. The appropriate DDL script for table creation is generated by VisualAge for Java into the persist class for the CMP bean. For more information about this script, see the VisualAge for Java release notes for the EJB Development Environment component.

1. In a process similar to that for setting up the administrative database, create the DB2/390 database, tablespace for the table, and tablespace for any BLOB fields in the schema for the CMP bean. (In DB2/390, BLOB fields are mapped to the LOB datatype, which is stored in a separate tablespace.)
2. Customize the DDL script provided by VisualAge for Java.
3. Using the customized DDL script, create the table for the CMP bean.
4. In the application server, deploy the CMP bean.

3.6: Interoperability with WebSphere Application Server for z/OS

Interoperability between the Advanced Edition (AE) product and WebSphere Application Server for z/OS can be achieved by using RMI over IIOP to propagate transaction and security contexts.

Transactions

WebSphere provides this function by supporting distributed transactions involving enterprise beans running in both the AE and z/OS application servers. To enable the interoperability, set the following flags in an AE server's JVM settings (system properties):

```
"-Dcom.ibm.ejs.jts.jts.ControlSet.nativeOnly=false" "-Dcom.ibm.ejs.jts.jts.ControlSet.interoperabilityOnly=true"
```

To set the arguments with the administrative console:

1. Select the application server node in the administrative domain tree.
2. Click the JVM Settings tab.
3. Add the arguments to the System Properties list.

Important: The default installation of the AE product provides a datasource that does not support the two-phase commit protocol. If your application involves a resource manager on a workstation and a resource manager on z/OS, you need to set up two-phase commit support. For this, you must use a JTA-enabled JDBC provider on the AE product by selecting the DB2 implementation class to be `COM.ibm.db2.jdbc.DB2XADataSource`. For details about defining a datasource and configuring it to use the JTA-enabled JDBC provider, see the related information.

Security authentication

In the Advanced Edition (AE) product, the security authentication service supports interoperability with WebSphere application servers for z/OS. If configured, this support allows only secure AE application servers to connect with secure z/OS targets that authenticate user IDs and passwords defined in a key file.

Important: This support requires the application of corrective service to the z/OS product. For details, see the AE product release notes.

To configure a secure application server to log on and access protected resources on a secure z/OS target, perform the following steps:

1. Create a key file and add log-in information
2. [Enable security with the AE application server](#)
3. [Enable security with the z/OS application server](#)
4. [Prepare a valid Java Key Store for the AE application server](#)
5. [Configure the AE application server](#)
6. [Configure the z/OS application server](#)

Create a key file and add log-in information

First, a key file must be created. Log-in information (target realm, user ID, and password) for each different z/OS target must be stored in the key file, which is simply a text file. When the security authentication service processes the key file, the passwords in the file will be encoded.

Add information to the key file in the following format:

```
RealmName UserID Password
```

Make sure that the data conforms to the following rules:

- One realm name, one user ID, and one password defined in each entry
- One entry per line
- No blank lines between entries
- Comments on separate lines only
- Begin any comment with a pound sign (#)

Example:

```
# Sample key file## First target realm#TargetRealm serverID serverPassword## Second target realm#TargetRealm2 serverID2 serverPassword2## End of key file
```

The realm name of a z/OS target is the IP name of the daemon as specified in the configuration of the z/OS product. The user ID and password are as defined for SSL-secured z/OS servers.

A sample file named `wsserver.key` contains these instructions and is shipped with the AE product. After installation, the sample file can be found in `AE_installation_root\properties`. You may use or modify the sample file as needed for testing.

Note: The key file can be placed anywhere on a host machine running the application server. However, it is recommended that you place the key file under a securable file system (for example, NTFS for Windows NT).

Enable security with the AE application server

Global security must be enabled. For details, see the related information.

Enable security with the z/OS application server

For details, see *WebSphere Application Server for z/OS and OS/390: Installation and Customization* (GA22-7834).

Prepare a valid Java Key Store for the AE application server

Secure Socket Layer (SSL) protocol is used to protect communication between AE and z/OS application servers. In order to complete the SSL connection at both ends, a valid Java key store for the AE server must be established as follows. You can use the Java key store file that is already used by the AE application server. If you do not have the key store file created already, see the related information to create one.

1. Extract the public key of the z/OS server by using the key management tool of z/OS.

For details, see *WebSphere Application Server for z/OS and OS/390: Installation and Customization* (GA22-7834).

2. From the AE product, start the iKeyman tool to open the key store file of the AE application server. Using the iKeyman tool, add the extracted public key from the z/OS server as a signer certificate into the AE server's Java store file. For efficient key management, it is recommended that you store the public key in the trust file.

For details, see the related information.

The AE server's Java key store file is now ready to interoperate with z/OS targets.

Configure the AE application server

After the Java key store file is ready and global security is enabled, resources like enterprise beans that will access z/OS also need to be configured. Before deploying the enterprise beans, RunAs Identity must be configured for each enterprise bean accessing z/OS resources. Because the security authentication service currently supports only AE application servers with z/OS targets, set RunAs Identity to `System Identity`.

Next, the server configuration must be updated. First, apply the Java key store file change as follows:

1. From the Administrator Console, select the application server.
2. From the right panel, click the **Services** tab.
3. Select **Object Request Broker** from the list.
4. Click **Edit Properties**.
5. From the Object Request Broker panel, click **Configure SSL**.
6. From the ORB SSL Configuration Properties panel, click the **Enable SSL** check box.
7. In the **Trust file name** field, type the absolute path for the previously created Java key store file. Provide the appropriate password and format for the trust file.
8. Click **OK** twice to go back to the administrative console.
9. Click **Apply**.
10. If the server is already running, restart it.

Next, set some Java properties as follows:

1. From the Administrator Console, select the application server.
2. From the right panel, click the **Services** tab.
3. Select **Object Request Broker** from the list.
4. Click **Edit Properties**.
5. Set the **Request timeout** and **Locate request timeout** properties to 0.

When the z/OS application server is first started, no server region is available for processing work. It is therefore recommended that you set these two properties to 0 in order to prevent potential timeouts.

6. Click **OK** twice to go back to the administrative console.
7. Click **Apply**.
8. If the server is already running, restart it.

Finally, add a property to the property file used by the application server (that is, `sas.server.props`). The `com.ibm.CORBA.keyFileName` property must be set to the absolute path of the login key file created earlier, as follows:

```
com.ibm.CORBA.keyFileName=c\:/WebSphere/AppServer/properties/wssserver.key
```

Configure the z/OS application server

For details, see *WebSphere Application Server for z/OS and OS/390: Installation and Customization* (GA22-7834).

3.7: Interoperability with Version 3.5.x

The interoperation of Version 3.5.x and Version 4.0 installations is supported only for versions 3.5.3 and later.

In order for Version 3.5.x installations to be able to access name servers on Version 4.0, an e-fix must be applied to each Version 3.5.x installation. This e-fix, PQ48255, is applied to the `ujc.jar` and `ns.jar` files of the Version 3.5.x installation. The e-fix is necessary only for versions 3.5.3 and 3.5.4; releases after 3.5.4 will not require the e-fix.

In order to use security between versions 3.5 and 4.0, the SSL protocol used in the Version 4.0 installation must be SSLv3. In the `sas.server.props` and `sas.client.props` files, verify that the following property has been set:

```
com.ibm.ssl.protocol=SSLv3
```

Then follow the instructions for applying security across domains in Article 6.6.18.1a.6. For additional information about SSL interoperability, see article 5.7.3.