Migration -- table of contents

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
1. Migrate or upgrade product prerequisites to supported versions As the product version changes, its prerequisites or corequisitesalso change. It is probably necessary to update your database, Webserver, JDK version, and other software.	Article 3.1
2. Migrate or upgrade to IBM WebSphere Application Server Version 4.0 In most cases, migration programs are available to ease the transition.However, some manual preparation may be necessary. Programmatic support for migrationfrom Version 2.0x is not provided. To migrate your installation from Version 2.0x,follow the documentation, starting at article 3.2.1.	Article 3.2.See alsoInstalling the product
3. Migrate administrative configurations If your company has been using a previous product version, the system administrator has probably fine-tuned various applicationand server settings for the environment. It is important to havea strategy for migrating these settings withmaximum efficiency and minimal loss.	Start with Article 3.2.2.
4. Update application code to supported specification and API levels **Note:**Support for several APIs has been removed in Version 4.0. 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.	Article 3.3
5. Redeploy applications on Version 4.0	Article 6.3.2

Before you begin, be sure to readTransitioning to Version 4.0.That article outlines the reorganization of the Version 4.0 productrelative 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 productrelative 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 usethe *Application Assembly Tool* (AAT)to create, edit, and view J2EE applications.

In Version 4.0, each application is installed into the server domainand 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. 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 ascollectively. Modules can be started collectivelyby either starting the application to which they belongor 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 stepsby 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 aministrative console, a Security Center enables you to perform all security tasks from a single location. This encompasseseverything 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 affectsthe 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 neededis 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.

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 rightlevel for the new installation. This driver will be required bythe product administrative server in order to connect to its administrativedatabase.

Migrating DB2, IBM HTTP Server, and other complimentary prerequisites

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

The compact disc version of the productincludes 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 sureyou download the installation package with the features you want.

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

Migrating non-IBM prerequisites

Some prerequisite or corequisite products, such as an Oracle or Sybase database, are not providedas 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 pageto determine which software requires migration or upgrade. Second, consult the documentation for the particular products to learn how to migrate to theversion supported by this product.

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

3.2: Migrating from previous product versions

Programmatic support for migrationfrom 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 upgrade Version 4.0 Advanced Single Server Edition with this release.For details, see article 3.2.3.

3.2.1: Migrating from Version 2.0x

Migration from Standard Edition Version 2.0x mustbe performed by hand.

Install product as new and migrate files and settings by hand

Uninstall Version 2.0x and start new withthis version, transferring application files and configuration settingsby 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 automatedmigration tool from Version 2.0x to Version 3.0x, the effort required to migrate to this versionby 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 laterby way of Version 3.0.After you have upgraded your Version 2.0x installationto Version 3.0 as specified in these instructions, install PTF 2 from the product Web site.At that point, you can use automated migration supportto upgrade the product to Version 3.5 or later.

For complete Version 3.0x installation and configuration information, consult the productWeb 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--
    - servlet.properties
    - admin_port.properties
    - rules.properties
    - jvm.properties
    - aliases.properties
    - conmgr.properties
    - 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 canperform 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.

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 isUpgradenode_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 readTransitioning to Version 4.0.That article outlines the reorganization of the Version 4.0 productrelative 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. Install the new version of IBM WebSphere Application Server.
4. Restore the configuration in the new installation.

Migrating administrative configurations

Tools for migrating administrative configurations are provided forversions 3.02 and later. This support enables either edition of Version 3.x to be upgraded to either Advanced or Advanced Single Server editionof 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 programdetects previously installed versions, collects informationabout 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 versionof the product and imports the backed-up administrative configuration.

Preinstallation phase

1. When you start the installation program, it automatically detects previous installed versions of the productand 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. 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.
4. When the installation program prompts you to check themigration 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 ofmigration 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 iswithin the Version 3.x directory structure, retain the migration backup directorybut 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 throughthe 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 configurationthat was exported in the first phase anddisplays the migration log file.

3.2.2.2: Migrating configurations manually

Manual migration might be necessary if either your current installationor 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 theoverall product migration process, as follows:

- Before upgrading the product, save the current configurationand 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 techniqueexports 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 toolsas 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 inthe 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 environmentinto 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 thesaved configuration and files. The directory is created if it doesnot already exist. This is a required parameter.
- currentWebSphereDirectory - The name of the installation root directory for the current Version 3.x installation. This can beeither a Standard or Advanced Edition installation. This is arequired parameter.
- administrationNodeName - The name of the administration node for thecurrently 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 nameand port number used by XMLConfig.
- [-traceString trace_spec [-traceFile file_name]] - These optional parameters are used to gather trace information for useby 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 aftermigration 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 (foundin the backup directory) with a Version 3.x XMLConfig tool to restore the previous configuration.The version of XMLConfig used by WASPreUpgrade encrypts passwordsduring 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 exportedby 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 exportthe current configuration.For more information about the XMLConfig tool,see the Version 3.x InfoCenter.

Before exporting an Advanced Edition configuration to a file, be sure that the administrative server is running. See XMLConfig documentation for further detailson 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 insteadof 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 procdure 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 reflectyour 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\ejs.jar;j:\websphere\appserver302\lib\console.jar;
j:\websphere\appserver302\lib\admin.jar;j:\websphere\appserver302\lib\repository.jar;
j:\websphere\appserver302\lib\sslight.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 mustupdate many of the names shown in thefollowing samples to reflect your configuration requirements:

```
j:\websphere\appserver\hosts >
j:\websphere\backup\websphere_3x_backup.userFiles\hostsj:\websphere\appserver\servlets >
j:\websphere\backup\websphere_3x_backup.userFiles\servletsj:\websphere\appserver\classes >
j:\websphere\backup\websphere_3x_backup.userFiles\classesj:\websphere\appserver\deployableEJBs >
j:\websphere\backup\websphere_3x_backup.userFiles\deployableEJBsj:\websphere\appserver\deployedEJBs
>   j:\websphere\backup\websphere_3x_backup.userFiles\deployedEJBsj:\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 toolsas 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 toolto 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 theVersion 3.x configuration to a Version 4.0 installation.

Creating and deploying J2EE applications

The J2EE programming model specifies an architecture for how applicationsare 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 J2EEapplications with the same name and deployed in the default server.All other Web resources and enterprise beans that are mapped but not includedin an enterprise application are mapped into a default J2EEapplication 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 recommendedthat you carefully analyze WASPostUpgrade.log(see the end of this topic), because detailed informationspecific to each bean deployed is saved in the log.

Security

Security settings that were applicable in the Version 3.x environmentare applied to J2EE security attributes as part of the migration process.LTPA security is not supported on Advanced Single Server Edition;this data cannot be migrated, so it is ignored.

Samples

Samples are not migrated; they have been updated specifically forJ2EE in Version 4.0. The new samples should be used instead of the onespreviously provided with Version 3.x product.

Mapping details

See the Related informationfor more specific information about how objects and attributesare mapped to the Version 4.0 configuration.

WASPostUpgrade parameters

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

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

- backupDirectoryName - The name of the directory that contains thesaved 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 theVersion 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 nodefor the current installation.
- [-configFile server_configuration_file] - If specified, this optional parameter can be used to specify the configuration file that will be updated. If this parameter is not specified, the file server-cfg.xml will be used. This parameter is used only if the configuration is beingrestored on an Advanced Single Server Edition installation.
- [-traceString trace_spec [-traceFile file_name]] - These optional parameters are used to gather trace information for use byIBM 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 viewedwith a text editor.

3.2.2.2.4: Mapping configurations to Version 4.0

This section details how objects and attributes are mappedto the Version 4.0 environment when you restore a configurationfrom an earlier product version.

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

  Because the typical location for these directories might include Version3.x installation directories and the location might be different in the new Version 4.0 installation, additional checking is donefor these entries if they are specified. Changed from Version 3.x, the default location forstdin, stdout, and stderr is the logs directory inVersion 4.0 installations. Existence of the passivation and working directories is checkedbefore 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 deployedas 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 thesebeans; 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 thedatasources 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 informationis defined at the container and EJB level and so must be mapped to theenterprise 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 thatis 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 differencefrom Version 3.x, in which many objects could be models and clones. All models andclones 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 aremapped as if they were not clones. Models that are not applicationserver 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 migratedto the one application server in the Version 4.0 environment.

- Node name

  A Version 3.x repositorycan 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 nonqualifiednetwork 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: Upgrading Version 4.0 Advanced Single Server Edition

The following process is applicable only if you have previously purchased and installedAdvanced Single Server Edition, Version 4.0:

1. If you have IBM HTTP Server installed,uninstall it.
2. Install the latest Advanced Single Server Edition product.If you uninstalled IBM HTTP Server in step 1, specify Custom installation in this step.Then select IBM HTTP Server and its plug-ins for installation with the application server.

The installation program backs up certain directories to a backup directory under the Version 4.0 installation root, which requires about 15MB of storage,and installs the new version into the same directory in which Version 4.0 was installed.

- On UNIX platforms, the backup directory is called *backup40*.
- On Windows platforms, the backup directory is called *was40aes_backup*.

All installed applications are backed up, their configurations are migrated,and they are deployed into the new application server installation.However, if you have made changes to the applications provided by IBM, see the following section.

Limitations to this process

- Upgrade of the Version 4.0 trial is not supported.
- The upgrade process backs up log and configuration files as well asthe following applications provided by IBM, if installed: admin, sampleApp, Samples, and petstore.Updated versions of these applications are then installed and deployed.Any user-modified bindings and mappings for the previous versionof these applicationsare not restored by the installation program.To restore bindings and document descriptor files for these applications, copy the files from the backup location.

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 transportwhen migrating to WebSphere Application Server Version 4.0.

There are two ways to migrate from OSE transportto 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 easierof the two methods.

1. Make sure that the Web server installationis at the supported level.If you are unsure what level of Web server product is supported, seethe 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 machineand 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 followingWeb 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 installationso 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 runon 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 installationso that it is in the location specified by the bootstrap.properties variablefor the Init directive in the obj.conf file.
5. Restart the Web server; you should be able to access applications that runon 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 installationso 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 runon 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 installationso 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 runon 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 oftechnologies for building powerful enterprise applications. Astechnology 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 currentlysupported by the product to provide additionalfeatures 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 theRelated 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
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	* Many JSP 1.0 applications will run unchanged in Version 4.0; however, some changes may be required or recommended. Version 2.0x only supported the JSP .91 Specification.
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	V2.0x supported JDBC 1.0 and a connection manager model. If still using Connection Manager, you mustswitch to connection pooling. Do not forget to switch to supported JDBC 2.0 drivers.
User profiles	IBM user profile APIs	No	Yes	Need to migrate from V2.0x deprecatedclasses for use with V4.0.
Sessions	IBM session support APIs	No	Yes	Need to migrate from V2.0x deprecatedclasses, changes to clustering, URL encodingfor use with V4.0.
Security	IBM security support	No	No	No action required.However, unlike previous versions, Version 4.0 does not protect URIsserved by an external Web server. Version 4.0 continues to protect URIs(including URIs for HTML files) that are served by the application server.
Transactions	Java 1.2 transactions support	Yes	Yes	Version 3.0x provided proprietary IBM packages to simulateJava 1.2 functionality. Version 2.0x did not provide anysupport. Migrate to Version 4.0 if your applications require thiskind of support.
XML configuration	XMLConfig tool	Yes	Yes	The XML Configuration Management Tool (XMLConfig) wasintroduced in Version 3.02.The DTD and many of the interfaces havechanged in Version 4.0.**XMLConfig is not supportedin 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.**WSCP is not supportedin 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.*javax.transaction.*`

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

  Any code using WebSphere Application Server data sources,including BMP entity beans and session beans that access databases, will needto 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.Repdeploy 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 binarycode compatible on a JDK 1.3 base. However, there are some incompatible cases. Forfurther details, see:

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

Important: The task of deploying enterprise beanshas changed significantly.Start at article 6.3 for an overview of theassembly and deployment process.

3.3.2: Migrating to supported Servlet specification and extensions

Servlets will require migration if they are not of the supported specificationlevel (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 HttpServiceResponse.callPage()

Calls to HttpServiceResponse.callPage() need to be replaced bycalls to RequestDispatcher, as shown.

Before -- Using HttpServiceResponse.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.HttpServiceRequest)req).setAttribute("message", message);
((com.sun.server.http.HttpServiceResponse)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 =
getServletContext().getRequestDispatcher("/Update.jsp");        rd.forward(req, res);
//((com.sun.server.http.HttpServiceRequest)req).setAttribute("message", message);
//((com.sun.server.http.HttpServiceResponse)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 ofthe XML for Java API, you can gain additional performanceby 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 ofa 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 APIis 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 elementand 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 andinsert 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 laterdiffers 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 itselfas 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 maynot be shared between contexts.This means that a session can no longer span Web applications. Objects added to a sessionby 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 containedthe 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 scripletto 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
dosetattribute set to *true*	JSP	JSP enabler	Assures that JSP session-scoped beans alwayscall setAttribute()
Write Contents option set to *Write all*	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 trackand manage sessions.

Migration from Version 2.0x

Relative to Version 2.0x, Version 3.0x introduced some changesto 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.
- 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 thebrowser making the HTTP request processes cookies. This differs from thebehavior in previous releases, which checked for the condition and haltedURL 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 introducedin IBM WebSphere Application Server Version 3.0x. Applications that use Version 3.0x connectionpooling need to be changed slightly and recompiled.

If existing applicationsare still using the connection manager model from Version 2.0x, you mustupdate the application code to use the currentconnection pooling model (see the Related information). The shift in models corresponds to a change in supported JDBCspecification levels.

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

Items newly deprecated in Version 4.0

The packages com.ibm.db2.jdbc.app.stdext.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 componentsthat use Version 3.0x connection pooling need to be changed slightly andrecompiled.First, replace the following import statement:

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

with this:

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

Connection pooling behavior in versions 3.5 and later changedrelative to that in Version 3.0x.If your application typically requirestwo or more connections to the same database manager,consider the multiple-connection scenarios inArticle 0.14.2.

3.3.8.2: Migrating servlets from the connection manager model

Servlets written to use the connection managermust 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	`import java.sql.*;  import com.ibm.servlet.connmgr.*;`	`import javax.sql.*;  import javax.naming.*;`
Modify servlet init() methods	`IBMConnSpec spec =  new IBMJdbcConnSpec("poolname", true,  "COM.ibm.db2.jdbc.app.DB2Driver", "jdbc:subprotocol:database",  "userid", "password");IBMConnMgr connMgr =  IBMConnMgrUtil.getIBMConnMgr();`	`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");` where *SampleDB* is the name of the data source. The WebSphere administrator provides informationon the arguments for the put() and lookup() methods.
Modifying how servlets obtain and close connections	`IBMJdbcConn cmConn =  (IBMJdbcConn)connMgr.getIBMConnection(spec);Connection conn = cmConn.getJdbcConnection();...cmConn.releaseIBMConnection();`	`Connection conn =  ds.getConnection("userid", "password");...conn.close();`
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
com.ibm.servlet.connmgr.IBMConnMgrUtil	• public static IBMConnMgr getIBMConnMgr() • public static IBMConnPoolSpec getPoolProperties(String poolName) • public static void addPoolProperties(IBMConnPoolSpec spec) • public static String urlToPoolName(String url)
com.ibm.servlet.connmgr.IBMConnMgr	• public IBMConnection getIBMConnection(IBMConnSpec connSpec) • public IBMConnection getIBMConnection(IBMConnSpec connSpec, String ownerClass)
com.ibm.servlet.connmgr.IBMConnection	• public boolean verifyIBMConnection() • public void removeIBMConnection() • public void releaseIBMConnection()
com.ibm.servlet.connmgr.IBMJdbcConn This class is derived from the IBMConnection class above and it implements one additional method, as shown.	• public Connection getJdbcConnection()
com.ibm.servlet.connmgr.IBMConnPoolSpec This class and the associated methods are intended for WebSphere Studio use only. Both methods are constructors.	• public IBMConnPoolSpec(String poolName, String poolType, int maxConnections, int minConnections, int connectionTimeOut, int maxAge, int maxIdleTime, int reapTime) • public IBMConnPoolSpec(String poolName, String poolType)
com.ibm.servlet.connmgr.IBMJdbcConnSpec The first three methods are constructors.	• 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 anapplication component obtains a connection to a database, then begins a transaction,the connection is closed automatically when the transaction ends. The connectionmust be obtained again before beginning another transaction.
- 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 fortwo-phase commit actions.