Migration -- table of contents

Product migration

API migration

Administrative configuration and database migration

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 3.5 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. Upgrade to IBM WebSphere Application Server Version 3.5 In most cases, migration programs are available to ease the transition.However, some manual preparation may be necessary.For example, Version 3.5 offers optionsfor backing up product customizations (such as Version 3.0x modifications to the Web server configuration file) before installing the Version 3.5 code, allowing preserved changes to be migrated to Version 3.5. To determine the version and release of your current installation,click **Help > About**from the menu bar of the Java administrative console. 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. Update application code to supported specification and API levels **Note:**Article 3.3.2a describes the migration issueswith using the Java Servlet API 2.2 specification. 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
4. Redeploy applications on Version 3.5	Article 6.4.2
5. 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.	Article 3.4

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 driver 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.

To migrate from Version 3.0x, you canuse the Migration Assistant or prepare the environment by hand.See article 3.2.2.

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 driver. `-trace` enables tracing.

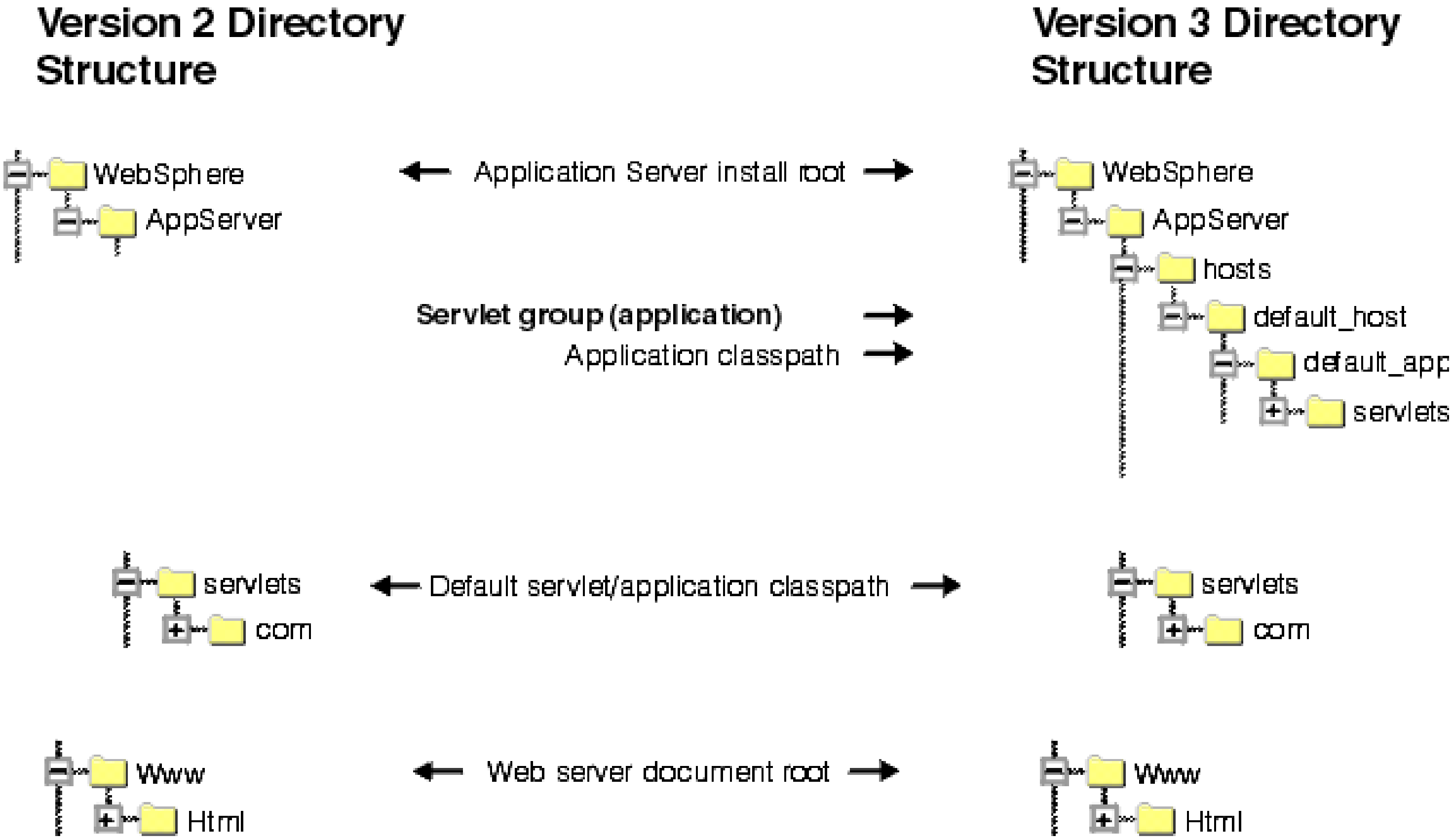
Running the tool gives you an ejscp script as the output. The output file name isUpgradenode_name.tcl. To complete the migration, run the script using ejscp with the following command:

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

3.2.1.2: Migrating Web application files from Version 2.0x directories

In addition to migrating Web applications to use supported APIs and specifications(see Related information), you need to move the actual application files fromthe directories in the previous version to the Version 3.5 directory structure.

The figure illustrates the Version 2 and Version 3 (including Version 3.5) directory structures for the default Web application (default_app).


Version 2 Directory Structure
Version 3 Directory Structure
WebSphere
AppServer
Application Server install root
WebSphere
AppServer
hosts
default_host
default_app
servlets
Servlet group (application)
Application classpath
servlets
com
Default servlet/application classpath
servlets
com
Www
Html
Web server document root
Www
Html

For Version 2.0x:

- Servlet and other application components were placed in the following paths:
    - <*AS_install_root*>\servlets
    - <*AS_install_root*>\classes
    - A user-defined reloadable classpath
- The static HTML and JSP files were placed in the Web server HTML document root

Optimal migration or quick migration

The above placement is not the optimal placement forVersion 3.5. However, a default_app is provided with Version 3.5in case you want to quickly migrate using a placement similarto that of Version 2.0x.

For instructions, select one of two Version-2.0x-to-Version-3.5 file migration paths in the Related information below. Again, the optimal path is the recommended one because withdrawal of support for the suboptimal path is imminent.

3.2.1.2.1: Optimally migrating Version 2.0x Web application files

To reorganize your Version 2 applications to take full advantage of the Version 3 Web application programming model, you should ultimately use the following migration method.

Task	Instructions
Use an administrative client to configure a Web application to include the servlets and Java components from the Version 2.0x Web application.	Article 6.6.8
Move the servlets and Java components to the class path of the newWeb application that you configured.	Article 6.4
Keep any servlets and Java components that were in the JVM class path (application server class path) in that location. These are servlets that you did not want to have reloaded on your Version 2 installation. The same restriction applies to Version 3.	Article 6.4
Move the static HTML and related resources to the new application document root directory.	Article 6.4.2
To enable serving of the HTML files, use one of the following methods: ● Add the SimpleFileServlet to the application. ● To enable serving of the JSP files, add the JSP compiler to the application. ● If needed, add other, optional internal servlets to the application. ● Add a pass rule for the document root to the Web server's configuration.	Article 4.2.1.2.3 Article 6.6.8.1.1 For the last method, consult the Web server documentation.

3.2.1.2.2: Quickly migrating Version 2.0x Web applications

For the fastest deployment, maintain the Version 2 file organization on your Version 3.5 installation:

1. Move the servlets and other Java components that were in the Version 2 \servlets directory to the Version 3 AS_install_root\servlets directory. If your servlets use package names, create subfolders under the \servlets directory to mirror the package name.
2. Move the servlets and Java components that were in another Version 2 reloadable class path to the Version 3 AS_install_root\servlets directory, or add the Version 2 reloadable class path to the Version 3 default_app application class path.
3. Keep any servlets and Java components that were in the system class path in that location. These are servlets that you do not want to have reloaded on your Version 2 installation. The same restriction applies to Version 3.
4. If your servlets or applications use .servlet configuration files, move those files to the same directory as the servlet.
5. Move the static HTML and related resources to the Web server document root.
6. Invoke the servlets and applications in the same manner you used with Version 2.

3.2.2: Migrating from Version 3.x

A summary of the product migration process follows.Much of this is done for you by the Migration Assistant.

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.

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 3.5, 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. Version 2.0x supported the Servlet 2.0 Specification andIBM extensions that were updated in Version 3.0x
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	Version 2.0x only supported the JSP .91 Specification. It is recommended you migrate to JSP 1.1.
JSP files	JSP 1.0 Specification	No**	Yes	Version 2.0x only supported the JSP .91 Specification. ** If you did not already migrate JSP .91 files for use with Version 3.x, it is recommended you migrate to JSP 1.1.
JSP files	JSP 1.1 Specification	No	not applicable	Version 2.0x only supported the JSP .91 Specification.

XML	XML 2.0.x supported XML 1.1.x supported with restrictions	No	No	Migration from 1.1.x to 2.0.x is notrequired, but you might decide to migrate basedon criteria and 1.1.x restrictions described in article 3.3.4.
JDBC and IBM database connection support APIs	JDBC 2.0; connection pooling model	No	Yes	V2.0x supported JDBC 1.0 and a connection manager model. If still using Connection Manager, it is recommended that you switch 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 V3.0x or V3.5
Sessions	IBM session support APIs	No	Yes	Need to migrate from V2.0x deprecatedclasses, changes to clustering, URL encodingfor use with V3.0x or V3.5
Security	IBM security support	No	No	No action required.
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 3.5 if your applications require thiskind of support.
XML configuration	XMLConfig tool	Yes	Yes	The XML Configuration Management Tool (XMLConfig) wasintroduced in Version 3.02. Some of the interfaces havechanged in Version 3.5.

3.3.2: Migrating to supported Servlet specification and extensions

Servlets will require migration if they are not of the supported specificationlevel (2.1) or they rely on deprecated or removed IBM servlet extensions.

See article 3.3.2a for migration considerations for WebSphereApplication Server Version 3.5.2.

Migrating to the supported Servlet specification

Refer to the Java Servlet API 2.1 specification for complete information concerning new and deprecated APIs. This table highlights a few of the new and deprecated classes and methods.

Recall, IBM WebSphere Application Server supported the Servlet2.1 Specification; if you already migrated servlets to 2.1 for use with thatrelease, no further action is required.

Article 4.2.1.2.1a describes the new Servlet 2.2 APIs.

Method or Class	Status and recommendation
RequestDispatcher	New. Use the forward method to forward a servlet response from one servlet to a second servlet for further processing. Use the include method to include part of the one servlet's response in the body of another servlet's response. Refer to the code example.
HttpSessionContext	Deprecated.
HttpSession.getSessionContext	Deprecated. For security reasons, no equivalent.
HttpSession.getMaxInactiveInterval	New. Sets the maximum time a session will be maintained by the servlet engine without a client request.
ServletRequest.getRealPath	Deprecated. Use ServletContext.getRealPath.
ServletContext.getServlet	Deprecated. Use ServletContext.getRequestDispatcher.
ServletContext.getResource	New. Use this method to obtain a servlet resource by requesting its URL.
ServletContext.getResourceAsStream	New. Use this method to obtain a servlet resource (as an InputStream) from its servlet context.
encodeUrl and encodeRedirectUrl methods of HttpServletResponse	Deprecated. But the fix is easy. Change `Url` to `URL` in the method names.
HttpSession.isRequestedSessionIdFromUrl	Deprecated. Another easy fix. Change `Url` to `URL` in the method name.

HttpServiceRequest.setAttribute()	Deprecated. See Migrating to supported JSP specification for details.
HttpServiceResponse.callPage()	Deprecated. Refer to the code example.

Migrating IBM extensions to the Servlet API

The following packages were are part of the Application Server Version 2.0x, but were removed or deprecated as of Version 3.

Method or Class	Status and recommendation
com.ibm.servlet.personalization.sam()	Removed -- no recommended replacement
com.ibm.servlet.servlets.personalization.util	Removed -- no recommended replacement
com.ibm.servlet.connmgr	Deprecated. Starting with Version 3.0x, IBM WebSphere Application Server has provided a built-in connection pooling function that eliminates the need for servlet programmers to use the connection manager APIs directly. Instead, servlets can be written to use the JDBC APIs to access the connection pool. You are encouraged to migrate servlets that use the deprecated connection manager APIs. See the Related information for details.

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);}}
```

3.3.2a: New Servlet Engine option for migrating applications to Servlet 2.2

WebSphere Application Server version 3.5.2 maintains compatibility withexisting applications while simultaneously supporting the Java Servlet API 2.2 specification.To ensure compatibility, a new option was added to *Servlet Engine* properties in the *Administrative console*.

This new option, the ***Select Servlet Engine Mode***, islocated on the Servlet Engine properties view. The ***Select Servlet Engine Mode*** option toggles between the following two different "runtime" modes:

- ***WebSphere Application Server 3.5 Compatibility Mode*** which maintainsbehavior with existing WebSphere Application Server V3.5 and V3.5.1 applications at the expense of full compliance with the Java Servlet API 2.2 specification.

  > **Note:** In ***compatibility mode***, the Servlet Engine is Servlet 2.2 specification level compliant except for the method andbehavior changes noted below. This capability is provided to allow existing WebSphere Application ServerV3.5 and V3.5.1 applications to successfully execute until they are migrated to fullycompliant Servlet 2.2 level applications.

- ***Full Servlet 2.2 Compliance Mode*** which maintains compliance with the Java Servlet API 2.2 specification at the expense of compatibility with existingWebSphere Application Server V3.5 and V3.5.1 applications.

The default mode is the ***Compatibility Mode***.You select your desired mode using the Administrative Console, Servlet Engine "General" tab.

Mode differences

The following table describes how the ***Select Servlet Engine Mode*** setting affects Servlet API methods and various behaviors.

> **Warning:** Specifying ***compliance mode*** for existingWebSphere Application Server V3.5 and V3.5.1 applications may generate incorrect results.

Type	Methods/ Behaviors	Compatibility mode	Compliance mode
API Methods	getCharacterEncoding() method	If the client request did not send any character encoding data, the default encoding of the server JVM is returned.	If the client request did not send any character encoding data, ***null*** is returned.
	getMimeType() method	If the file extension does not map to a valid mime type, the mime type ***www/unknown*** is returned.	If the file extension does not map to a valid mime type, ***null*** is returned.

Behaviors	Default content type on *response buffer reset*	On *response buffer reset*, the content type of the request is reset to ***text/html***.	On *response buffer reset*, the content type is cleared and and not set to a default value.
	HTTP Session scoping	Values placed in the HTTP Session object have a **global scope**, across all Web applications.	Values placed in the HTTP Session object have a **scope limited to the Web application** that created the value.
	Request mapping behavior	● Exact mapping is not supported. ● Wildcard mapping is an *implied* wildcard. That is, **/Servlet** really means **/Servlet***. ● Any URL pattern specified without **/*** on the end is assumed to be a wildcard rule, and **/*** is added in the Servlet runtime. ● Any URL pattern provided with **/*** on the end is accepted and used as is.	● The servlet specification pattern mapping logic is followed, including support for exact matches. ● To specify the URL, the Servlet 2.2 specification allows the following syntax: 1. A string beginning with **/** and ending with **/*** specifies a wildcard match. 2. A string beginning with ***.** specifies an extension mapping. 3. All other strings are used as exact matches. ● The Servlet 2.2 specification indicates how requests for resources are mapped to the appropriate resources. Mapping occurs in the following order: 1. exact match 2. longest wildcard match

			3. matching extension 4. default servlet (defined by **/ URL**).
	Auto-Invoker	The default invoker's URL is: `/servlet/`	The default invoker's URL is: `/servlet/*`

3.3.3: Migrating to supported JSP specification

If using JSP 1.0 already, no action is required, with one exception, Version 3.5 does not have Bean Scripting Framework (BSF) support for using LotusXSL and other scripting languages in JSP 1.0 files. Version 3.0x applications depending on this support will require modification.

Version 3.5.2 supports the Bean Scripting Framework (BSF) for Netscape'sRhino JavaScript language. See article 4.2.5 for moreinformation.

Version 3.5.2 also supports JSP 1.1 and continues to support JSP 1.0 and JSP .91 files.See article 3.3.2a for migration details.

IBM WebSphere Application Server Version 3.5 continues to support the JSP .91 Specification, but it is recommended that you migrate applications containing JSP .91 files to JSP 1.0, if you have not already done so.

See the Related information for the migration steps required to useJSP .91 files with Version 3.5, and tips for migrating themto JSP 1.0 instead.

3.3.3.1: Updating JSP .91 files for use with Version 3.5

If using JSP .91 files or servlets that you have not already been using with Version 3.0x, and the JSP .91 files or servlets cast to either of these methods:

- com.sun.server.http.HttpServiceRequest
- com.sun.server.http.HttpServiceResponse

either replace the deprecated calls **or** recompile the files.

Additionally, if migrating JSP .91 files last used with IBM WebSphere Application Server Version 1.x, you need to eliminate <REPEATGROUP> tags. See below for details.

Option 1: Modify deprecated calls

The tables summarize calls to the deprecated HttpServiceRequest and HttpServiceResponse classes,and provide replacement code.

Before:	com.sun.server.http.HttpServiceRequest.setAttribute()
After:	javax.servlet.http.HttpServletRequest.setAttribute()

A code example is provided to showhow to migrate to RequestDispatcher:

Before:	com.sun.server.http.HttpServiceResponse.callPage()
After:	javax.servlet.RequestDispatcher

Option 2: Recompile files

As an alternative to replacing the deprecated calls to HttpServiceRequest andHttpServiceResponse, recompile your JSP .91 files or servlets developed for Application Server Version 2.0x before using them with Application Server Version 3.5.

Recompiling is necessary because starting with Version 3.0x, HttpServiceRequest and HttpServiceResponse are provided as interfaces (instead of classes) that are implemented by the WebSphere servlet engine.

If you do not recompile the servlets or JSP files, the Java Virtual Machine (JVM) will crash on Windows NT systems due to a suspected bug in the JDK.

It ispossible that you already recompiled the files for use with Version 3.0x. Insuch a case, it is not necessary to compile them again for Version 3.5.

Migrating JSP .91 files from IBM WebSphere Application Server Version 1.x

The Application Server Version 1.x supported an additional tag, <REPEATGROUP> for repeating a block of HTML tagging for data that is already logically grouped in the database. Because this release does not support the <REPEATGROUP> tag, remove that tag from any JSP files that you want to use on the Application Server Version 3.5.

3.3.3.2: Tips for migrating JSP .91 files to JSP 1.0

Referring to WebSphere example code for the purposes of illustration,the tips below cover some main steps in migrating JSP .91 to JSP 1.0.

Replacing <SERVLET> with <jsp:include>

Use the JSP 1.0 equivalent of <SERVLET> to include data in a JSP page from another file.

Example	CounterServletOutputPage.jsp
JSP .91	`<SERVLET CODE="WebSphereSamples.Counter.CounterServlet"></SERVLET>`
JSP 1.0	`<jsp:include page="/servlet/WebSphereSamples.Counter.CounterServlet" />`
Discussion	The CounterServletOutputPage.jsp file and the servlet it invokes are part ofthe Version 3.5 Web application named WSsamples_app, with the Web Application Web Path setting "/WebSphereSamples." Using a WebSphere administrative client to view the WSsamples_app Web application, you will find that it contains the Auto-Invoker servlet, which enables you to call servlets by classname. Specified within the Auto-Invoker servlet is the Servlet Web Path List,which has the single entry "default_host/WebSphereSamples/servlet." Now, the JSP and CounterServlet servlet are both under the Web Application Web Path of /WebSphereSamples. Relative to /WebSphereSamples, the servlet needs the additional qualifier of /servlet to properly locate it (as specified in the Auto-Invoker). This results in the "/servlet/WebSphereSamples.Counter.CounterServlet" in the <jsp:include>tag.

Replacing <BEAN> with <jsp:useBean>

Use the JSP 1.0 equivalent of <BEAN> to make an existing or newly created bean available from within the JSP file.Four variations are possible.

Variation 1: JSP is to create the bean

Example	PollServletInputPage.jsp
JSP .91	`<BEAN NAME="getQuestionDBBean" TYPE="WebSphereSamples.Poll.GetQuestionDBBean" CREATE="YES" INTROSPECT="YES" SCOPE="request"></BEAN>`
JSP 1.0	`<jsp:useBean id="getQuestionDBBean"type="WebSphereSamples.Poll.GetQuestionDBBean" class="WebSphereSamples.Poll.GetQuestionDBBean"scope="request"/>`
Discussion	You no longer have the explicit attribute of CREATE="YES". Instead, if the bean with the name specified by the id attribute is not found within the specified scope, then an instance of bean will be created according to the class attribute. JSP NAME attribute corresponds to the JSP 1.0 id attribute. It is no longer an INTROSPECT attribute. (The JSP .91 scope of requests and sessions carry over to JSP 1.0, plus some new ones for JSP 1.0.) Compare with variation 1 with variation 2.

Variation 2: JSP is to use existing bean

Example	PollServletResultPage.jsp
JSP .91	`<BEAN NAME="pollQueryDBBean" TYPE="WebSphereSamples.Poll.PollQueryDBBean" CREATE="NO" INTROSPECT="NO" SCOPE="request"></BEAN>`
JSP 1.0	`<jsp:useBean id="pollQueryDBBean" type="WebSphereSamples.Poll.PollQueryDBBean" scope="request"/>`
Discussion	Compare variation 2 with variation 1, which creates a bean if one does not exist. JSP 1.0 version no longer has the class attribute from JSP .91. If a bean instance corresponding to the id attribute not found in the specified scope, there will be an error. As a result, the bean will not be created.

Variation 3: Properties are to be set for bean

Example	CenterGeneric.jsp
JSP .91	`<BEANNAME="getQuestionDBBean"TYPE="WebSphereSamples.YourCo.Poll.GetQuestionDBBean" CREATE="YES" INTROSPECT="NO" SCOPE="request"> <PARAM NAME="userID" VALUE="wsdemo"></BEAN>`
JSP 1.0	`<jsp:useBeanid="getQuestionDBBean" type="WebSphereSamples.YourCo.Poll.GetQuestionDBBean" class="WebSphereSamples.YourCo.Poll.GetQuestionDBBean" scope="request" /><jsp:setProperty name="getQuestionDBBean"     property="userID"     value="wsdemo" />`
Discussion	The example above has been shortened somewhat to set only one parameter. Note that with JSP .91, the <PARAM> tag used within the <BEAN> tag. In JSP 1.0, you must instead use the <jsp:setProperty> tag outside of<jsp:useBean> tag. You must properly link setting the property of an existing bean by the name attribute within <jsp:setProperty> pointing to the bean identified by the id attribute within <jsp:useBean>. Similar considerations for <jsp:getProperty>.

Variation 4: Invoke methods on a bean

Example	FeedbackServletResultPage.jsp

JSP .91	`<% try { java.lang.String _p0_1 = feedbackQuery.getWSDEMO_FEEDBACK_NAME(0); %>`
JSP 1.0	No change from JSP .91 to JSP 1.0
Discussion	The NAME attribute in <BEAN> tag and id attribute in <jsp:useBean> tag are equivalent. Both identify a bean named feedbackQuery. For either JSP specification, invoking a method on a bean is identical.

Incorporating IBM extensions to JSP 1.0

See the Related information for references of IBM tags that extend JSP 1.0. The tags might give you additional ideas for replacing JSP .91 tagging functionality. In some cases, IBM extensions provide functionality thatwas removed in the switch from JSP .91 to JSP 1.0. The remainder of this article focuses on one such tag, <tsx:repeat>.

Replacing <REPEAT> with <tsx:repeat>

<tsx:repeat> provides for repeating information, which is useful in creating HTML tables. <REPEAT> from JSP .91 is usually used with <INSERT> to actually insert data fromspecified bean. <REPEAT> from JSP .91 does not have an equivalent in the JSP 1.0 specification. However, the IBM extension <tsx:repeat> provides much the same function.

Example	timeout.jsp (available in Advanced Ediiton only)
JSP .91	`<REPEAT INDEX="i">  <%timeoutBean.getBalance(i);%>    <TD><INSERT BEAN="timeoutBean" PROPERTY="balance"></INSERT></TD></REPEAT>`
JSP 1.0	`<txt:repeat index="i">  <TD> <%= timeoutBean.getBalance(i) %> </TD></txt:repeat>`
Discussion	Note that there is an actual call, using Java syntax, of getBalancemethod of timeoutBean, within loop of <tsx:repeat>, rather than use of IBM JSP 1.0 extension <tsx:getProperty>. This is because getBalance method requires an explicit argument to specify which row of data fromunderlying array in timeoutBean is to be returned. Thus, <tsx:getProperty>not suitable.

3.3.4: Migrating to supported XML API

If your XML applications use XML for Java API Version 1.1.x, you must migrate them to API Version 2.0.x.

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 following table summarizes the methods of the API Version 1.1.x class com.ibm.xml.parser.Parser that are not supported or implemented in the API Version 2.0.x:

Method	Status
addNoRequiredAttributeHandler	Not supported. Throws java.lang.IllegalArgumentException.
getReaderBufferSize	Not supported. Throws java.lang.IllegalArgumentException.
setErrorNoByteMark	Not supported. Throws java.lang.IllegalArgumentException.
setProcessExternalDTD	Not implemented.
setReaderBufferSize	Not supported. Throws java.lang.IllegalArgumentException.
setWarningNoDoctypeDecl	Not implemented.
setWarningNoXMLDecl	Not implemented.
stop	Not implemented.

The following table summarizes Version 1.1.x methods that are deprecatedin Version 2.0.x of the com.ibm.xml.parser package:

Deprecated method	Recommendation
EntityDecl.getName()	Use getNodeName().
EntityDecl.getNDATAType()	Use getNotationName().
EntityDecl.isNDATA()	Do not use.
Namespace.getUniversalName()	See createExpandedName().
Parent.addElement(Child)	Use appendChild().
TXAttribute.getUniversalName()	Use createExpandedName().
TXAttribute.setAttribute(TXAttribute)	Use setAttributeNode().
TXElement.getName()	Use getNodeName() or getTagName().
TXElement.getUniversalName()	Use createExpandedName().
TXElement.isEmpty()	See hasChildNodes().
TXNotation.getName()	Use getNodeName().
TXNotation.setName(String)	Do not use.
TXText.splice(Element, int, int)	Do not use.

3.3.5: Migrating to supported user profile APIs

Migrating from Version 3.x

Changes to code are not required.

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 3.0x

If your existing applications already use Version 3.0x session support, no code changes are required.

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.

- 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, it is recommended that you update 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.

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 managershould continue to work in the Application Server Version3.5 environment, provided that the servlets use a subset of the connection manager APIs thatare deprecated but still supported. See the Related information for the API subset, which is anticipated to cover most existing servlets.

For most servlets, the migration consists of simple code changes. Because you should not write new servlets using 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.ejs.ns.jndi.CNInitialContextFactory");Context ctx = new InitialContext(parms);DataSource ds (DataSource)ctx.lookup("jdbc/sample");` 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 deprecated in the Application Server Version 3.5 environment and might not work with releases beyond this one. You should not write new servlets using the connection manager. Instead, write new servlets using the connection pooling model from Version 3.5.

3.3.8.3: Deprecatedconnection 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 continue to be supported. The classes are now deprecated, so the details of connection manager coding are not discussed.

Deprecatedconnection manager class	Methods
com.ibm.servlet.connmgr.IBMConnMgrUtil The last three of the four methods are intended for IBM WebSphere Studio use only.	● 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 3.0x of the product ran with a 1.1 level of JDK. Version 3.0x included packages written by IBM to provide transaction support features usually provided by JDK 1.2. Now that Version 3.5 runs with JDK 1.2, applications should no longer import the proprietary IBM packages, but instead import the open Java 1.2 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 1.2.

Other transaction considerations for Version 3.5:

- 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.

3.3.10: Migrating to supported XML configuration

Some interfaces of the XMLConfig tool have changed.The changes in XML descriptions of the elements aresummarized in the following table:

Element	Changed syntax
jdbc-driver	Now supports <install-info/> and <uninstall-info/> elements (optional)
session-manager	<session-manager name=*session manager name*></session-manager>
userprofile-manager	<userprofile-manager name=*user profile manager name*></userprofile-manager>

Changes from Version 3.0x

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.

3.4: Migrating administrative configurations

There are two ways to migrate from Version 3.x:

- Use the Migration Assistant.
- Manually complete the same steps as the Migration Assistant would.This might be necessary for nonstandard installations.

For details, see the Related information.If you choose the automated method,consider reading the associated article about manual migrationto learn more about the migration process.

3.4.1: Using the Migration Assistant

The Migration Assistant helps you migrate from IBM WebSphere Application Server Version 3.0.2x to Version 3.5x. To use the WebSphere Migration Assistant, you must have WebSphere Application Server installedat least at the 3.0.2x level. If you do not, you must upgrade to that level before continuing. Also, if you installed WebSphere Application Server V3.0.2 by using the native installation program, ensure that the JAVA_HOME variable is set correctly in the setupCmdLine file before continuing.

Starting the Migration Assistant

To start the Migration Assistant on Windows NT/2000, run the migration executable (migration.exe). If you have a CD, migration.exe is in the \nt directory.

To start the Migration Assistant on UNIX:

1. Log onto your machine with superuser (root) privileges.
2. Run the migration script file (`./migration.sh`), which is in the /cdrom directory.

Note: The Migration Assistant does not check for the bootstrap port on which the current installationis running. If the bootstrap port is something other than the default (900),migration will fail. To prevent failure,start the Migration Assistant with the`-nameServerPort` option, specifying the appropriate port.

What the Migration Assistant does

The Migration Assistant detects if you have Version 3.0.2.x installed. If a version earlier than Version 3.0.2.x is installed, the Migration Assistant tells you to upgrade to Version 3.0.2.x before continuing and points you to this InfoCenter for more information.

If the Migration Assistant detects that Version 3.0.2.x is installed, the wizard provides a series ofpanels that walk you through migration to Version 3.5, including:

- Exporting, saving and restoring your previous administrative configuration
- Uninstalling Version 3.0.2.x
- Testing your system for the right prerequisites
- Installing the upgraded version of WebSphere Application Server

The Migration Assistant also leads you to information about the final steps of installing the WebSphereApplication Server, such as migrating application components to APIs and specifications.

If you exit the Migration Assistant before completing all of the steps on the panels and later restart the Migration Assistant, the wizard restarts where you left off.

Details on the migration.exe and its command-line arguments for Windows

The migration.exe program, which runs on Windows NT/2000, determines if a migration should be performed in the course of an installation of Version 3.5.

migration.exe can be run with no arguments, which is what happens you run theVersion 3.5 installation program. When executed with no arguments, migration.exe makes a number of checks against values stored in the Windows NT/2000 system registry:

- If Version 3.0.2 or later is present in the registry, it performs a `migration add_run` command and then launches the migration wizard with the `migration uncond` command.
- If a version previous to 3.0.2 but after 3.0 is present in the registry, it displays the **Unsupported Version**

panel and then launches the installation program setup.exe.

- If Version 3.0 is present in the registry, it performs a `migration add_run` command and then launches the migration wizard with the `migration uncond` command.
- If a version previous to 3.0 is present in the registry, it displays the **Unsupported Version** panel and then launches the installation program setup.exe.
- If no version of IBM WebSphere Application Server is in the registry, it launches the installation program setup.exe.

migration.exe may also be run with either the `uncond`, `add_run`, or `remove_run` arguments:

`migration uncond`

Causes the migration wizard to be launched, without checking for an installed Application Server product and without checking the version of the current Application Server product.

`migration add_run`

Verifies that the current directory is the migration directory, checking for either migration.exe or setup.exe. If the current directory is verified as the migration directory, the program stores this directory under the registry keys:

```
"SOFTWARE\\IBM\\WebSphere Application Server - Migration Assistant""Home"
```

`migration add_run` primarily adds an auto-run key to the registry. The auto-run key is *WebSphere migration Assistant* and its run command is `migration.exe uncond`.

`migration remove_run`

Causes the auto-run key which was added to the system registry by the `migration add_run` command to be removed, causes the migration directory to be removed from the system registry, and causes files in the migration directory to be deleted.

3.4.2: 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, export the current XML configurationand back up necessary files.
- After upgrading the product, restore the configuration.

Exporting the current administrative configuration

Before exporting the configuration to a file, be sure an administrative server is running.

A sample export command for Version 3.0x follows.You may have to update many of the values used in this sample to reflect your configuration requirements.

```
j:\jdk1.1.8.orig\bin\javaDserver.root=j:\websphere\appserver302Dcom.ibm.CORBA
.ConfigURL=file:/j:/WebSphere/AppServer302/properties/sas.client.propsclasspath 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 900export j:\websphere\backup \websphere_302_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\appserver302\hosts > j:\websphere\backup\userFiles\hosts
j:\websphere\appserver302\servlets > j:\websphere\backup\userFiles\servlets
j:\websphere\appserver302\classes > j:\websphere\backup\userFiles\classes
j:\websphere\appserver302\deployableEJBs > j:\websphere\backup\userFiles\deployableEJBs
j:\websphere\appserver302\deployedEJBs > j:\websphere\backup\userFiles\deployedEJBs
j:\websphere\appserver302\properties > j:\websphere\backup\programFiles\properties

Also copy one of the following, depending on your operating system:

- For Windows NT/2000:

  j:\websphere\appserver302\bin\admin.config > j:\websphere\backup\bin\admin.config
- For Netware:

  j:\websphere\appserver302\bin\setupCmdLine.ncf > j:\websphere\backup\bin\setupCmdLine.ncf
- For AIX and Solaris:

  j:\websphere\appserver302\bin\setupCmdLine.sh > j:\websphere\backup\bin\setupCmdLine.sh

Restoring the configuration in the new installation

First, copy the backed-up configuration files into the new installation directory. Be sure to update any names shown in the following samples to reflect your configuration requirements.

j:\websphere\backup\userFiles\hosts > j:\websphere\appserver35\hosts
j:\websphere\backup\userFiles\servlets > j:\websphere\appserver35\servlets
j:\websphere\backup\userFiles\classes > j:\websphere\appserver35\classes
j:\websphere\backup\userFiles\deployableEJBs > j:\websphere\appserver35\deployableEJBs

Restore the following file:

j:\websphere\backup\bin\admin.config > j:\websphere\appserver35\bin\admin.config

In addition, restore one of the following files, depending on your operating system:

- For Windows NT/2000:

  j:\websphere\backup\bin\setupCmdLine.bat > j:\websphere\appserver35\bin\setupCmdLine.bat
- For Netware:

  j:\websphere\backup\bin\setupCmdLine.ncf > j:\websphere\appserver35\bin\setupCmdLine.ncf
- For AIX and Solaris:

  j:\websphere\backup\bin\setupCmdLine.sh > j:\websphere\appserver35\bin\setupCmdLine.sh

Next, import the configuration.

A sample import command for Windows NT/2000 follows.Remember that you must update many of the values to reflect your configuration requirements.

```
j:\websphere\appserver35\jdk\jre\bin\JavaDserver.root=j:\websphere\appserver35Dcom.ibm.CORBA.ConfigURL=file:/j:\websphere\appserver35/properties/sas.client.propsclasspath
j:\websphere\appserver35\lib\ibmwebas.jar; j:\websphere\appserver35\lib\servlet.jar;
j:\websphere\appserver35\lib\xml4j.jar;j:\websphere\appserver35\lib\ujc.jar;
j:\websphere\appserver35\lib\ejs.jar;j:\websphere\appserver35\lib\console.jar;
j:\websphere\appserver35\lib\admin.jar;j:\websphere\appserver35\lib\repository.jar;
j:\websphere\appserver35\lib\sslight.jar;j:\websphere\appserver35\lib\tasks.jar; j:\websphere\appserver35\properties
com.ibm.websphere.xmlconfig.XMLConfig-adminNodeName cally-nameServiceHost cally-nameServicePort 900import
j:\websphere\backup\websphere_302_backup.xml
```