



Migrating, coexisting, and interoperating

Note

Before using this information, be sure to read the general information under Appendix C, "Notices," on page 155.

Compilation date: November 30, 2004

© Copyright International Business Machines Corporation 2004. All rights reserved.

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

Contents

How to send your comments	vii
Part 1. Product overview	1
Chapter 1. What is new in this release	3
Chapter 2. How do I migrate, coexist, and interoperate?	5
Chapter 3. Overview and new features for migrating, coexisting, and interoperating	11
Overview of migrating, coexisting, and interoperating	12
Deprecated features in Version 6.0	12
Deprecated features in V6.0	13
Deprecated features in V5.1.1	16
Deprecated features in V5.1	17
Deprecated features in V5.0.2	20
Deprecated features in V5.0.1	21
Deprecated features in V5.0	22
Migrating and coexisting	23
Migration and coexistence overview.	23
Configuration mapping during migration	25
Migrating WebSphere programming model extensions (PMEs)	30
Part 2. Migrating	33
Chapter 4. Migrating applications	35
Web applications.	35
Migrating Web application components from WebSphere Application Server Version 4.x	35
Migrating Web application components from WebSphere Application Server Version 5.x	37
Migrating HTTP sessions	38
Migrating enterprise bean code to the supported specification	38
Migrating enterprise bean code from Version 1.0 to Version 1.1	39
Migrating enterprise bean code from Version 1.1 to Version 2.1	39
Migrating Apache SOAP Web services to Web Services for J2EE standards.	40
Migrating to Version 3 of the UDDI Registry.	43
Using a remote database for the UDDI Registry	46
Migrating a Version 4.0 data access application to Version 6.0	46
Converting a 2.2 Web module to a 2.3 Web module.	47
Converting a 1.1 EJB module to a 2.1 EJB module (or later)	48
Add the EJB modules and Web modules to an EAR file	48
Installing the Application on WebSphere Application Server	48
Connection considerations when migrating servlets, JavaServer Pages, or enterprise session beans	49
Mail migration tip	50
Migrating security configurations from previous releases	50
Migrating custom user registries	51
Migrating trust association interceptors	54
Migrating Common Object Request Broker Architecture programmatic login to Java Authentication and Authorization Service.	56
Migrating from the CustomLoginServlet class to servlet filters	59
Migrating Java 2 security policy	60
Migrating Version 5 Application Profiles to Version 6.	62
Migrating internationalized applications	63

Chapter 5. Migrating product configurations	65
Stand-alone migrations	66
Migrating V4.0.x of WebSphere Application Server to a V6 stand-alone Application Server	66
Migrating V4.0.x of WebSphere Application Server to a remote stand-alone V6 machine	68
Migrating V5.x Express to V6 Express	70
Migrating V5.x of WebSphere Application Server to a V6 stand-alone Application Server on a remote machine	71
Migrating V5.x of WebSphere Application Server to a V6 Application Server	73
Migrating a V5.x Application Server configuration instance to a V6 Application Server profile	73
Migrating from an operating system that is no longer supported	74
Configuring WebSphere Application Server after migration	75
XML parser for Java code	76
Configuring WebSphere Application Server for DB2 access	78
Chapter 6. Migrating Web server configurations	81
Chapter 7. Migrating administrative scripts	83
Migrating V4.0.x administrative scripts to V6 wsadmin	83
Example: Migrating - Creating an application server	85
Example: Migrating - Starting an application server	85
Example: Migrating - Installing an application	85
Example: Migrating - Installing a JDBC driver	87
Example: Migrating - Creating a server group	87
Example: Migrating - Stopping a node	88
Example: Migrating - Stopping an application server	88
Example: Migrating - Pinging running servers for the current state	88
Example: Migrating - Regenerating the node plug-in configuration	89
Example: Migrating - Testing the DataSource object connection	89
Example: Migrating - Cloning a server group	90
Example: Migrating - Enabling security	90
Example: Migrating - Disabling security	91
Example: Migrating - Modifying the virtual host	91
Example: Migrating - Modifying and restarting an application server	91
Example: Migrating - Removing an application server	92
Example: Migrating - Modifying the embedded transports in an application server	93
Example: Migrating - Connecting to a remote server	93
Example: Migrating - Uninstalling an application	93
Example: Migrating - Editing an application	93
Example: Migrating - Modifying attributes of application servers, applications, and other configured objects	94
Example: Migrating - Displaying help	94
Example: Migrating - Listing actions available for configured objects	94
Example: Migrating - Setting the server trace specification	95
Migrating administrative scripts from 5.x to 6.0	95
Example: Migrating - Allowing configuration overwrite when saving a configuration	97
Part 3. Coexisting	99
Chapter 8. Coexisting	101
Coexistence support	101
Setting up Version 4.0.x and Version 6 coexistence	103
Setting up Version 5 and Version 6 coexistence	104
Setting up Version 6 coexistence	105
Chapter 9. Configuring ports	107
Port number settings in WebSphere Application Server versions	107

Part 4. Interoperating	109
Chapter 10. Interoperating	111
Chapter 11. Container interoperability.	115
Chapter 12. Web Services-Interoperability Basic Profile.	119
Chapter 13. Interoperability issues for security	121
Chapter 14. Interoperating with a C++ common object request broker architecture client	123
Chapter 15. Interoperating with previous product versions	125
Chapter 16. JNDI interoperability considerations	127
Chapter 17. Application profiling interoperability	129
Chapter 18. Interoperating with asynchronous beans	131
Chapter 19. Interoperating with schedulers	133
Appendix A. Migration tools	135
The clientUpgrade command	135
The convertScriptCompatibility command	136
WASPreUpgrade command	137
WASPostUpgrade command	140
Using the Migration wizard	143
Migrating from V4 to V6 with the Migration wizard	143
Migrating a V5 Application Server to a V6 Stand-alone Application Server with the Migration wizard	146
Appendix B. Troubleshooting migration	149
Migration utility troubleshooting tips	152
Migrating a previously non-root configuration to root	153
Appendix C. Notices	155
Appendix D. Trademarks and service marks	157

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
 1. Display the article in your Web browser and scroll to the end of the article.
 2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
 3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-0206.

Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Part 1. Product overview

Chapter 1. What is new in this release

This topic provides a high level overview of what is new and changed in Version 6. The audience is anyone who is evaluating or planning to use Version 6, particularly users of previous versions or editions.



Welcome to Version 6, which helps you:

- Rapidly understand and obtain value from new J2EE 1.4 technology
- Make the most of your existing resources and lower your total cost of ownership
- Work more productively, with easier to use installation and administration interfaces
- Support your growing business with a firm foundation to the WebSphere platform

There is now greater consistency in the WebSphere Application Server family from top to bottom, including in the Express version. Such features include support for J2EE 1.4 across the family, which makes it easier to develop and deploy applications using industry standard tools, and support for the latest Web Services standards, making it easier to integrate applications inside the enterprise as well as externally with customers, partners and supplier.

IBM WebSphere Application Server Version 6 provides customers better scaling so more concurrent users can access an application built on the software. This can help reduce administrative and licensing costs for companies and provides better flexibility to quickly bring additional user resources on board.

What is new for migration

A new migration wizard collects data for the migration command line tools. The wizard also monitors and reports migration status. Note that the installation program no longer performs the migration.

See “Using the Migration wizard” on page 143 for additional information.

Another big change is the introduction of separate installation routines for the Application Server product, the Web server, the Web server plug-ins, and the clients so that you can install only what you need on a particular machine.

At a glance, these are the main new items in installation, migration, coexistence, and interoperability:

- Simplified operating system setup and prerequisites
- Updated Launchpad application for planning and starting installation more easily
- Improved documentation, including diagrams of installation scenarios
- More componetized installation procedures let you install only what you need
- Improved installation logging and status indicators
- Smaller footprint
- Binary files are shared by installations on the same machine
- Updated First Steps application for getting started quickly after installation
- Simplified service and fixes
- Easier removal and reinstallation
- New migration wizard

For additional details about each installation item, including links to relevant documentation, see the *Getting Started* PDF.

What is new for administrators



The biggest improvements and changes in system administration, monitoring, and tuning can be summarized as follows:

- Changes to the default configuration
- Fine grained application update and other deployment improvements
- Enhanced administrative infrastructure, with J2EE 1.4 related changes
- Growing set of administrative commands
- Improved installation and configuration, with profiles
- Improved monitoring and performance tuning
- Default messaging provider
- Even more administrative improvements!

For details about each item, including links to relevant documentation, see the *Administration: Operations* PDF.

Improvements to this documentation

- New How do I?... feature gathers documentation, multimedia demonstrations, tutorials, cheat sheets, and presentations for commonly performed tasks
- Improved installation documentation
- Improved scripting documentation
- Conceptual overviews and what is new topics have better content and are easier to find
- Quick start shortcuts direct you quickly to key materials
- New *Learn about WebSphere applications* section for studying the programming model
- Improved, task-based table of contents
- Improvements to troubleshooting content and placement

Chapter 2. How do I migrate, coexist, and interoperate?

- Migrate product and Web server configurations
- Migrate your applications
- Coexist
- Interoperate
- Troubleshoot migration



Legend for "How do I?..." links

Documentation 	Show me 	Tell me 	Guide me 	Teach me
Refer to the detailed steps and reference	Watch a brief multimedia demonstration	View the presentation for an overview	Be led through the console pages	Perform the tutorial with sample code
Approximate time: Varies	Approximate time: 3 to 5 minutes	Approximate time: 10 minutes+	Approximate time: 1/2 hour+	Approximate time: 1 hour+

Migrate product and Web server configurations

Expand **Migration > Migrating product configurations** in the information center navigation to obtain instructions for various navigational paths to Version 6.0.x from previous versions and other editions.



Obtain an overview of migration and coexistence options

Find out about supported migration paths, migration tools, the goals of migration and coexistence, and more.

Documentation

Tell me



Determine what configuration information needs to change

Learn what changes during migration, which involves migrating a single instance to another single instance on the same machine or a separate machine.

Documentation



Review the software and hardware prerequisites

Learn which product configurations are supported.

Documentation



Locate migration tools

Become acquainted with the tools that the product provides for automated migration.

Documentation



Tell me



Migrate Web server plug-in configurations

Migrate the configurations used by the application server to communicate with supported Web servers.

Documentation



Migrate your applications

Expand **Migration > Migrating WebSphere applications** in the information center table of contents for information about migrating specific application components, their deployment descriptors, and their administrative configurations (such as EJB container settings).



Identify deprecated features that you might be using

Review a summary of deprecated features in Version 6.0.x. As they become available, links to additional information are provided to help you migrate away from deprecated features.

Documentation






Review the software and hardware prerequisites

Learn which product configurations are supported. Specification levels are included.

Documentation:
Supported hardware

and software 



Learn about WebSphere applications

Use this section as a starting point to investigate the technologies used in and by applications that you deploy on the application server.

Documentation:
"Learn about
WebSphere
applications" in the
*Developing and
Deploying Applications*

PDF 

Coexist

Expand **Migration > Coexisting** in the information center navigation to obtain instructions for various coexistence scenarios. Coexistence, as it applies to WebSphere Application Server products, is the ability of multiple installations of WebSphere Application Server to run on the same machine at the same time. Multiple installations include multiple versions and multiple instances of one version. Coexistence also implies various combinations of Web server interaction.



Review supported coexistence scenarios

All combinations of V4.x products, V5.x products, and V6.0 products can coexist so long as there are no port conflicts. There are some coexistence limitations for V5 products that have the embedded messaging feature installed, as described in the documentation. The installation wizard looks for certain existing installations to determine if it should prompt for coexistence information.

Documentation:
Chapter 8,
"Coexisting," on page

101 



Configure ports for coexistence or interoperability

This topic provides reference information about identifying port numbers in versions of WebSphere Application Server, as a means of determining port conflicts that might occur when you intend for an earlier version to coexist or interoperate with Version 6.0.x.

Documentation: “Port number settings in WebSphere Application Server versions” on page 107



Obtain valid port settings for coexistence

Version 6 autonomically detects free ports among profiles installed from the same core product files. The port mechanism does not detect ports in use by previous versions or by other products.

You must verify that you are using ports that do not conflict with other products and programs running on your node. Select free ports on ports selection panel of the installation wizard or the Profile creation wizard.

Documentation:

- See the *Getting Started* PDF for more information.
- “Port number settings in WebSphere Application Server versions” on page 107



Interoperate

Interoperability is exchanging data between two coexisting product installations.



Interoperate across product versions

WebSphere Application Server, Version 6.0.x is generally interoperable with some previous versions. However, there are specific requirements to address for each version. Certain changes are required to support interoperability between versions.

Documentation:
Chapter 10,
“Interoperating,” on

page 111



Configure ports for coexistence or interoperability

This topic provides reference information about identifying port numbers in versions of WebSphere Application Server, as a means of determining port conflicts that might occur when you intend for an earlier version to coexist or interoperate with Version 6.0.x.

Documentation: “Port
number settings in
WebSphere
Application Server
versions” on page 107



Troubleshoot migration

Troubleshoot problems that occur during migration.



Troubleshoot migration problems

Find details about logs and messages pertaining to the automated migration tools, plus other tips for solving problems that occur during migration.

Documentation:
Appendix B,
“Troubleshooting
migration,” on page

149



Chapter 3. Overview and new features for migrating, coexisting, and interoperating

This topic summarizes the contents and organization of the migration documentation, including links to conceptual overviews and descriptions of new features.



- “Overview of migrating, coexisting, and interoperating” on page 12

Sections in the migration documentation:



“Deprecated features in Version 6.0” on page 12

This topic summarizes deprecated programming interfaces as of Version 6.0.



Chapter 5, “Migrating product configurations,” on page 65

This topic describes how to migrate administrative configurations somewhat manually, with the migration tools. Consider using the automated Installation wizard instead.



Chapter 6, “Migrating Web server configurations,” on page 81

This topic describes what to do to migrate a Web server from supporting IBM WebSphere Application Server, Version 4.0.x or Version 5.x.x, to support IBM WebSphere Application Server, Version 6.0.



Migrating WebSphere applications

This topic provides migration, coexistence, and interoperability instructions that are specific to various types of applications. For example, you can focus on moving Web applications with the latest supported specifications; migrating Web services from Apache SOAP; or integrating two types of messaging support.



Chapter 7, “Migrating administrative scripts,” on page 83

This topic provides guidance for migrating from WebSphere Application Server V4.0 wscp scripts to wsadmin in V6.x.



Chapter 8, “Coexisting,” on page 101

This topic is a starting point for finding information about which coexistence scenarios are supported, and how to set them up.



Chapter 10, “Interoperating,” on page 111

This topic describes how to interoperate across product versions.



Chapter 9, “Configuring ports,” on page 107

This topic is a starting point for finding information about configuring ports, particularly in coexistence scenarios.



Appendix B, “Troubleshooting migration,” on page 149

This topic describes how to identify and handle a variety of problems that are encountered during migration, interoperability, and coexistence activities.

Overview of migrating, coexisting, and interoperating

This topic provides links to conceptual overviews of migrating applications and their environment as well as coexisting and interoperating various versions or editions of the application serving environment.

“Migration and coexistence overview” on page 23

This topic describes common product topologies that you can install with WebSphere Application Server, Version 6 products.



Presentations from Education on Demand

- Migration overview

Deprecated features in Version 6.0

This document summarizes deprecated features in Version 6.0. As they become available, links to additional information will be provided to help you migrate away from deprecated features.

IBM Deprecation policy

A deprecated class or interface is supported for at least two full product releases or three full years (whichever is longer) before being removed from the product entirely.

Deprecation list

The following tables summarize what is deprecated, by release and version. Use these tables to calculate the earliest release in which the item might be removed, according to the deprecation policy.

Each table represents on what release/version the deprecation took effect. Each table describes what is being deprecated, for example: features, APIs, scripting interfaces, tools, wizards, publicly exposed configuration data, naming identifiers, constants, etc., and, where possible, the recommended migration action IBM suggests.

This article consists of the following versions/releases:

- Deprecated features in V6.0
- Deprecated features in V5.1.1
- Deprecated features in V5.1
- Deprecated features in V5.0.2
- Deprecated features in V5.0.1
- Deprecated features in V5.0

Deprecated features in V6.0

Application programming model and container support features																
<p>Support for the following tsx tags in the JSP engine is deprecated:</p> <ul style="list-style-type: none">• repeat• dbconnect• dbquery• getProperty• userid• passwd• dbmodify																
<p>Recommended migration action:</p> <p>Instead of using the tsx tags, you should use equivalent tags from the JavaServer Pages Standard Tag Library (JSTL). JSTL is supported in WebSphere Application Server V6.0, and the tag library is shipped with the product. Use the following list as a guideline for converting tsx tags to JSTL tags:</p>																
<table><thead><tr><th>tsx tag</th><th>JSTL tag</th></tr></thead><tbody><tr><td>tsx:repeat</td><td>c:forEach</td></tr><tr><td>tsx:dbconnect</td><td>sql:setDataSource</td></tr><tr><td>tsx:dbquery</td><td>sql:query</td></tr><tr><td>tsx:getProperty</td><td>Use standard EL syntax, for example, c:out value="\${book.title}", where book is the current index in the result set</td></tr><tr><td>tsx:userid</td><td>Use the <i>user</i> attribute of the setDataSource tag</td></tr><tr><td>tsx:passwd</td><td>Use the <i>password</i> attribute of the setDataSource tag</td></tr><tr><td>tsx:dbmodify</td><td>sql:update</td></tr></tbody></table>	tsx tag	JSTL tag	tsx:repeat	c:forEach	tsx:dbconnect	sql:setDataSource	tsx:dbquery	sql:query	tsx:getProperty	Use standard EL syntax, for example, c:out value="\${book.title}", where book is the current index in the result set	tsx:userid	Use the <i>user</i> attribute of the setDataSource tag	tsx:passwd	Use the <i>password</i> attribute of the setDataSource tag	tsx:dbmodify	sql:update
tsx tag	JSTL tag															
tsx:repeat	c:forEach															
tsx:dbconnect	sql:setDataSource															
tsx:dbquery	sql:query															
tsx:getProperty	Use standard EL syntax, for example, c:out value="\${book.title}", where book is the current index in the result set															
tsx:userid	Use the <i>user</i> attribute of the setDataSource tag															
tsx:passwd	Use the <i>password</i> attribute of the setDataSource tag															
tsx:dbmodify	sql:update															
Application services features																
<p>The WebSphere JRAS Extensions API is deprecated in this release. No further enhancements are planned for JRAS support.</p>																
<p>Recommended migration action:</p> <p>Begin moving over to the java.util.logging package (JSR47), particularly for any new code you are writing.</p>																
<p>The UDDI version 2 EJB interface to the UDDI Registry is deprecated.</p>																
<p>Recommended migration action:</p> <p>There is no replacement for the EJB interface. This interface is included in WebSphere Application Server V6.0 for compatibility with V5.x. Users do not need to take any specific actions, and can continue to use the version 2 EJB API, but should be aware that it does not include any UDDI functionality that is new to UDDI version 3, and that the interface may be removed in a future release of WebSphere Application Server.</p>																

The UDDI4J version 2 class library, `uddi4jv2.jar`, is deprecated.

Recommended migration action:

Start using the version 3 UDDI APIs. A client library is provided to simplify constructing and sending UDDI v3 requests from Java. This is the IBM UDDI v3 Client for Java, provided in `uddiv3client.jar`. The UDDI4J APIs may still be used, but you should be aware that they do not provide access to any of the new UDDI version 3 functionality, and that they may be removed in a future release of WebSphere Application Server.

All of the low-level UDDI Utility Tools (UUT) APIs, such as `BusinessStub`, `ServiceStub`, etc., are deprecated. These are all replaced by the high-level `PromoterAPI` interface.

Recommended migration action:

Start using the `PromoterAPI` in place of these low-level APIs, which will be removed in a future release of WebSphere Application Server. The `PromoterAPI` provides the same functionality at a higher level of abstraction.

The following methods in the J2EE Connector Architecture runtime are deprecated:

- `com.ibm.ws.management.descriptor.xml.ConnectionFactory.xml` (`getPoolContents` and `getAllPoolContents` methods)
- `com.ibm.websphere.j2c.ConnectionManager` interface
- `com.ibm.websphere.j2c.ConnectionEventListener` interface

Also, container-managed authentication aliases on a J2C Connection Factory or Datasource are deprecated.

Recommended migration action:

- `getPoolContents` and `getAllPoolContents` replaced by `showPoolContents` and `whoAllPoolContents`
- `ConnectionManager` interface replaced by J2EE Connector Architecture 1.5 `LazyAssociatableConnectionManager` interface
- `ConnectionEventListener` interface replaced by J2EE Connector Architecture 1.5 `LazyEnlistableConnectionManager` interface.

For container-managed authentication aliases, specify the container-managed credentials via the application's resource binding information.

The `ApplicationProfile` property on the `WorkManager` panel in the administrative console is deprecated.

Recommended migration action:

None.

Two items from the `DataSource` panel in the administrative console are deprecated:

- Container-Managed Authentication Alias
- `DefaultPrincipleMapping`

Recommended migration action:

None.

All classes in the `com.ibm.websphere.servlet.filter` package are deprecated:

- `ChainedRequest`
- `ChainedResponse`
- `ChainerServlet`
- `ServletChain`

Recommended migration action:

Re-architect your legacy applications to use `javax.servlet.filter` classes rather than `com.ibm.websphere.servlet.filter` classes. Starting from the Servlet 2.3 specification, `javax.servlet.filter` classes give you the capability to intercept requests and examine responses. They also allow you to achieve chaining functionality, as well as embellishing and/or truncating responses.

MIME filtering is deprecated. MIME filters were first introduced in WebSphere Application Server V3.5 as a way for servlets to embellish, truncate, and/or modify the responses generated by other servlets, based on the MIME types of the output content.

Recommended migration action:

`javax.servlet.filters`, which were introduced in the Servlet 2.3 specification, allow users to plug in filters which can intercept requests to and responses from servlets. They also have the capability to modify content flowing in either direction.

`javax.servlet.filters` maintain all the functionality of MIME filters. `javax.servlet.filters` are standard APIs, and are supported by all compliant application servers. Refer to the Servlet 2.3 specification for more information.

Container-managed persistence (CMP) entity beans configured with method level access intent may run into data access problems, like deadlock. Therefore, the method level access intent is deprecated.

Recommended migration action:

Re-configure CMP entity beans to use bean level access intent, or re-configure Application profiles with WebSphere Application Server Tool (AST).

All the methods and fields in `com.ibm.websphere.product.product` and `com.ibm.websphere.product.buildInfo` classes are deprecated. Hence, the following methods from `com.ibm.websphere.product.WASProduct` class (which involves `com.ibm.websphere.product.product` and `com.ibm.websphere.product.buildInfo` objects) are deprecated:

- `public product getProductByFilename(String basename)`
- `public product getProductById(String id)`
- `public boolean productPresent(String id)`
- `public boolean addProduct(product aProduct)`
- `public boolean removeProduct(product aProduct)`
- `public Iterator getProducts()`
- `public Iterator getProductNames()`
- `public String loadVersionInfoAsXMLString(String filename)`
- `public String getProductDirName()`
- `public static String computeProductDirName()`

Recommended migration action:

Use the following supported methods from `com.ibm.websphere.product.WASDirectory`:

- `public WASProductInfo getWASProductInfo(String id)`
- `public boolean isThisProductInstalled(String id)`
- `public WASProductInfo[] getWASProductInfoInstances()`
- `public String getWasLocation()`

Also, instead of getting product information (name, version, build level, build date) from the old `WASProduct` API (`com.ibm.websphere.product.WASProduct`), you should now use the following methods in the `WASDirectory` class to get that information:

- `com.ibm.webpsphere.product.WASDirectory.getName(String)`
- `com.ibm.webpsphere.product.WASDirectory.getVersion(String)`
- `com.ibm.webpsphere.product.WASDirectory.getBuildLevel(String)`
- `com.ibm.webpsphere.product.WASDirectory.getBuildDate(String)`

Data Access Beans, which are shipped with WebSphere Application Server in `databeans.jar`, are deprecated.

Recommended migration action:

Instead of using Data Access Beans, you should use Service Data Objects (SDO).

Security features

SOAP-Security (XML digital signature) based on Apache SOAP implementation is deprecated.

Recommended migration action:

Instead of using SOAP-Security, you should migrate your application to JSR-109 implementation of Web service. Also, migrate (reconfigure your application) to use WSS (Web Service Security) 1.0 implementation.

WSS (Web Service Security) draft 13 specification-level support is deprecated in favor of the WSS 1.0 implementation.

Recommended migration action:

Applications should be migrated to the supported WSS 1.0 standard. The draft-level support does not provide interoperability with some third party vendors, as the message level has been changed between the draft and the WSS 1.0 implementation.

WSS 1.0 is only supported in J2EE 1.4 applications. Hence, you need to migrate applications to J2EE 1.4 first. The next step is to use AST/RAD tooling to reconfigure WSS for the migrated application. There is no automatic migration of WSS in this release of AST/RAD tooling for V6.0; the migration has to be done manually.

The following SPI has also been deprecated:
`com.ibm.wsspi.wssecurity.config.KeyLocator`

You need to migrate your implementation to the new SPI for WSS 1.0 support in V6.0:
`com.ibm.wsspi.wssecurity.keyinfo.KeyLocator`

Finally, the JAAS LoginModule implementation needs to be migrated to the new programming model for JAAS LoginModule in V6.0.

System administration features

Configuring resources under cell scope is deprecated.

Recommended migration action:

You should configure resources under cluster scope instead. In previous releases, you configured cell scope resources to allow the cluster members to share the resource configuration definition. In Version 6, cell scope resource configuration is discouraged because cell scope resources are visible to every node in the cell, even though not every node in the cell is able to support the resource.

The `depl.extension.reg` and `installdir` options for the `install` command in the AdminApp scripting object are deprecated.

Recommended migration action:

There is no replacement for the `depl.extension.reg` option. In V5.x, this option was a no-op. For the `installdir` option, use the `installed.ear.destination` option instead.

Performance features

The PMI Client API, which was introduced in V4.0 to programmatically gather performance data from WebSphere Application Server, is deprecated.

Recommended migration action:

The Java Management Extension (JMX) interface, which is part of the J2EE specification, is the recommended way to gather WebSphere Application Server performance data. PMI data can be gathered from the J2EE-managed object MBeans, or from the WebSphere PMI Perf MBean. While the J2EE MBeans provide performance data about a specific component, the Perf MBean acts as a gateway to the WebSphere Application Server PMI service, and provides access to the performance data for all the components.

Deprecated features in V5.1.1

Application services features

The following JDBC drivers are deprecated:

- MS SQL Server 2000 Driver for JDBC
- SequeLink JDBC driver for MS SQL Server

Recommended migration action:

If you are using either of these JDBC drivers and still want to use MS SQL Server as their database, you can switch to the Connect JDBC driver. You can purchase the Connect JDBC driver from DataDirect Technologies, or you can use the Connect JDBC driver shipped with WebSphere Application Server, which is free for use with WebSphere Application Server.

Deprecated features in V5.1

Installation and migration tools
<p>The Application Assembly Tool (AAT) used for developing J2EE applications is being replaced by the Assembly Tool (AT) component of the Application Server Toolkit (AST).</p> <p>Recommended migration action:</p> <p>Instead of running the Application Assembly Tool, users will install and run the Assembly Toolkit component of the Application Server Toolkit. AST is based on the eclipse framework. Upon starting the AST, the J2EE function is found by opening the J2EE Perspective.</p>
<p>JDOM (a Java representation of an XML document which provides an API for efficient reading, manipulating and writing documentation). The currently packaged version of JDOM in WebSphere Application Server will not be packaged in future releases of WebSphere.</p> <p>Recommended migration action:</p> <p>Go to JDOM and get the latest copy of JDOM and bundle it inside your application. Note: Customers running WebSphere Studio Application Developer Integration Edition Version 4.1 applications will need to migrate them to WebSphere Studio Application Developer Integration Edition Version 5.0.</p>
<p>In future releases, IBM intends to remove the C++ Object Request Broker (ORB), the C++ library for IDL valuetypes and the WebSphere Application Server C++ security client. IBM will no longer ship or support the Common Object Request Broker Architecture (CORBA) C++ Developer Kit. The CORBA technology is a bridge for migration to a Java 2 Platform Enterprise Edition (J2EE) and WebSphere Application Server environment.</p> <p>In addition to the preceding information, the CORBA C++ client feature will be removed from the Application Clients installation image in future releases.</p> <p>Recommended migration action:</p> <p>It is recommended that customers migrate to the Object Request Broker (ORB) service for Java technology that ships with WebSphere Application Server. However, there is no equivalent J2EE functionality for the C++ security client or the C++ Valuetype library. Customers that require such functionality must provide or develop their own.</p> <p>The deprecation of the CORBA C++ Developer Kit does not affect support for CORBA interoperability with vendor software for CORBA services. View the following links for additional information about interoperability:</p> <ul style="list-style-type: none">• CORBA Interoperability Samples documentation• IBM WebSphere Application Servers CORBA Interoperability white paper
<p>IBM Cloudscape 5.1.x.</p> <p>Recommended migration action:</p> <p>No action is required at this time.</p>
Servers and clustering features

IBM HTTP Server (IHS) 1.3.x.

Recommended migration action:

If you are using IHS 1.3.x with modules:

- that are shipped as part of IHS 1.3.x packages, you do not need to take any action to migrate those modules.
- supplied by a third party (including other IBM products), you need to obtain IHS/Apache 2 versions of these modules from the third party.
- that have been customized or are in-house, you need to port these modules to the new IHS/Apache 2 API.

Application programming model and container support features

Bean Scripting Framework (BSF). JSP execution and debugging functionality is being deprecated in WebSphere Application Server version 5.1.

Recommended migration action:

If using the Javascript, Tcl, and Python languages, the functionality will need to be re-architected. If using BSF scripting in your own custom applications, they will be unaffected. Custom scripts written for the WebSphere Application Server admin console will also be unaffected.

This functionality will continue to exist in WebSphere Application Server app server release 5.1, and succeeding releases, until version 6.0. If debugging JSPs you may have to restart the app server during Javascript debugging sessions.

Data access programming interfaces in com.ibm.websphere.rsadapter.

Relational resource adapter interface:

(com.ibm.websphere.rsadapter)

Methods have been deprecated in these types:

```
com.ibm.websphere.rsadapter.OracleDataStoreHelper
    public void doSpecialBlobWork(ResultSet rset,
        InputStream[] data, String[] blobColumnNames)
    public String assembleSqlString(String[] blobColumnNames,
        StringBuffer whereClause, String[] varValues, String tableName)
```

Recommended migration action:

These relational resource adapter deprecated methods do not impact the application.

Note: You will not need to implement these deprecated methods in their subclasses if you have the subclass of OracleDataStoreHelper class. Those deprecated methods will not be called by the WebSphere Application Server runtime.

Webcontainer API modifications:

Note: There are no declared deprecations. The only changes are caused because of a Java API that changed between 1.3 and 1.4.

The changed class is `com.ibm.websphere.servlet.error.ServletErrorReport`. The return signature for `getStackTrace()` is changed because `java.lang.Throwable` now defines the same method with a different return signature.

- Old method signature

```
public String getStackTrace();  
// returns a String representation of the  
exception stack
```

- New method signature (JDK 1.4, WebSphere Application Server 5.1)

```
public StackTraceElement[] getStackTrace();  
// returns an array of stack trace  
elements
```

- Replacement method (WebSphere Application Server 5.1) (a replacement method that carries on the old functionality has been provided):

```
public String getStackTraceAsString();  
// returns a String representation  
of the Exception Stack
```

Recommended migration action:

If you are using `com.ibm.websphere.servlet.error.ServletErrorReport.getStackTrace()` and expecting a return type of `String`, you need to change your application to use the replacement method.

Application services features

Data access binaries -- Common Connector Framework:

The following jar files will be deprecated in V5.1:

- `ccf.jar`
- `ccf2.jar`
- `recjava.jar`
- `eablib.jar`

Recommended migration action:

The J2EE Connector Architecture solution should be used instead of the Common Connector Framework.

Setting the XA partner log directory via the 'TRANLOG_ROOT' variable is deprecated in V5.1.

Recommended migration action:

The setting currently stored in the `TRANLOG_ROOT` variable (if any) will need to be added to the Transaction Service panel for any servers who wish to use the XA partner log. If the default location is to be used, then no action is required. The Transaction Service panel can be found on the Administrative Console by selecting Application Servers on the left, choosing the application server to be modified, and selecting Transaction Service on the panel that is displayed. The directory currently in `TRANLOG_ROOT` should be entered in the Logging Directory box on the panel.

Security features

Security programming interfaces:

- The API is being deprecated for

```
com.ibm.websphere.security.auth.WSPrincipal.getCredential()
```

Recommended migration action:

Instead of getting the WSCredential from the principal, you should now use one of the following methods to get the Subject which contains the WSCredential:

- The RunAs Subject is the Subject used for outbound requests.
- The Caller subject is the Subject that represents the authenticated caller for the current request.
- The methods to use to get the runAs and caller subjects are

```
com.ibm.websphere.security.auth.WSSubject.getRunAsSubject()
```

and

```
com.ibm.websphere.security.auth.WSSubject.getCallerSubject()
```

respectively.

- The interface is being deprecated in

```
com.ibm.websphere.security.auth.WSSecurityContext
```

Recommended migration action:

Use JAAS for any authentication related functionality.

- The exception is being deprecated in

```
com.ibm.websphere.security.auth.WSSecurityContextException
```

Recommended migration action:

Use JAAS for any authentication related functionality.

- The class is being deprecated in

```
com.ibm.websphere.security.auth.WSSecurityContextResult
```

Recommended migration action:

Use JAAS for any authentication related functionality.

System administration features

The following class is deprecated:

```
com.ibm.websphere.rsadapter.DB2390DataStoreHelper
```

Recommended migration action:

If you currently use the DB2390DataStoreHelper class for the DB2 Legacy CLI-based provider when you are accessing data, you should now use the DB2DataStoreHelper class.

If you currently use the DB2390DataStoreHelper class for the DB2 Universal JDBC provider driver when you are accessing data, you should now use the DB2UniversalDataStoreHelper class.

Deprecated features in V5.0.2

Application programming model and container support features

Apache SOAP channel in Web services gateway.

Recommended migration action:

Gateway services should be deployed to the SOAP HTTP channel instead of the Apache SOAP channel. The Endpoint (URL) of the service will be different for this channel and therefore client programs that are talking to the gateway will need to use the new service Endpoint.

Apache SOAP, WEBSJAVA.SOAP:

- soap.jar,
- wsssoap.jar

Application services features

Data access programming interfaces in com.ibm.websphere.rsadapter.

Relational resource adapter interface:

(com.ibm.websphere.rsadapter)

Methods have been deprecated in these types:

- com.ibm.websphere.rsadapter.DataStoreHelper
public int processSQL(java.lang.String.sqlString, int isolevel,
boolean addForUpdate, boolean addextendedForUpdateSyntax);
public DataStoreAdatperException mapException(DataStoreAdapterException e);
- com.ibm.websphere.rsadapter.GenericDataStoreHelper
public int processSQL(java.lang.String.sqlString, int isolevel,
boolean addForUpdate, boolean addextendedForUpdateSyntax);
public DataStoreAdatperException mapException(DataStoreAdapterException e);
- com.ibm.websphere.rsadapter.WSCallHelper
public static DataStoreHelper createDataStoreHelper(String dsClassName)

Recommended migration action:

These relational resource adapter deprecated methods do not impact the application.

Note: You will not need to implement these deprecated methods in their subclasses if you have the subclass of GenericDataStoreHelper. Those deprecated methods will not be called by the WebSphere Application Server runtime.

For com.ibm.websphere.rsadapter.WSCallHelper, please use the getDataStoreHelper(datasource) method to get a DataStoreHelper object.

System administration features

The **testConnection** command in the AdminControl scripting object (\$AdminControl TestConnection configId props) is deprecated. Running this command in WebSphere Application Server, Version 5.0.2 or later returns the following message: WASX7390E: Operation not supported - testConnection command with config id and properties arguments is not supported. Use testConnection command with config id argument only.

Recommended migration action:

As of WebSphere Application Server, Version 5.0.2 or later, the preferred way to test a data source connection is the testConnection command passing in the data source configuration ID as the only parameter.

The **getPropertiesForDataSource** command in the AdminControl scripting object (\$AdminControl getPropertiesForDataSource configId) is deprecated. This command incorrectly assumes the availability of the configuration service when you run it in the connected mode. Running this command in WebSphere Application Server, Version 5.0.2 or later returns the following message: WASX7389E: Operation not supported - getPropertiesForDataSource command is not supported.

Recommended migration action:

There is no replacement for this command.

Deprecated features in V5.0.1

Application services features

Data access programming interfaces in com.ibm.websphere.rsadapter.

Relational resource adapter interface:

(com.ibm.websphere.rsadapter)

Methods have been deprecated in these types:

- com.ibm.websphere.rsadapter.DataStoreHelper
public int processSQL(java.lang.String sqlString, int isolevel);
- com.ibm.websphere.rsadapter.GenericDataStoreHelper
public int processSQL(java.lang.String sqlString, int isolevel);
- com.ibm.websphere.rsadapter.DB2390DataStoreHelper
public int processSQL(java.lang.String sqlString, int isolevel);

Recommended migration action:

These relational resource adapter deprecated methods do not impact the application.

Note: You will not need to implement these deprecated methods in their subclasses if you have the subclass of com.ibm.websphere.rsadapter.GenericDataStoreHelper. Those deprecated methods will not be called by the WebSphere Application Server runtime.

Deprecated features in V5.0

Application services features

The following three methods from com.ibm.websphere.appprofile.accessintent.AccessIntent are deprecated:

```
public boolean getPessimisticUpdateHintWeakestLockAtLoad();  
public boolean getPessimisticUpdateHintNoCollision();  
public boolean getPessimisticUpdateHintExclusive();
```

This is a base api.

Recommended migration action:

Rather than using the three deprecated methods on the AccessIntent interface, developers should use the following method from the same interface:

```
public int getPessimisticUpdateLockHint();
```

The possible return values are defined on the AccessIntent interface:

```
public final static int PESSIMISTIC_UPDATE_LOCK_HINT_NOCOLLISION = 1;  
public final static int PESSIMISTIC_UPDATE_LOCK_HINT_WEAKEST_LOCK_AT_LOAD = 2;  
public final static int PESSIMISTIC_UPDATE_LOCK_HINT_NONE = 3;  
public final static int PESSIMISTIC_UPDATE_LOCK_HINT_EXCLUSIVE = 4;
```

Web application programming interfaces -- Various Version 5 methods in com.ibm.websphere.ServletErrorReport

Performance features

Performance Monitoring Infrastructure -- Various Version 5 public methods in:

- com.ibm.websphere.pmi.stat.StatsUtil

Recommended migration action:

There is no replacement for StatsUtil.

- com.ibm.websphere.pmi.PmiJmxTest

Recommended migration action:

Use PmiClient.findConfig().

- com.ibm.websphere.pmi.client.PmiClient

Recommended migration action:

The getNLSValue (String key) is replaced by getNLSValue (String key, String moduleID).

Migrating and coexisting

This topic describes migrating, which is copying the configuration from a previous release of a WebSphere Application Server product into a new release. This topic also describes coexisting, which is running a new release of a WebSphere Application Server product on the same machine at the same time as you run an earlier release, or running two installations of the same release of a WebSphere Application Server product on the same machine at the same time.

If you have a previous version, you must decide whether to copy the configuration and applications of the previous version to the new version. Migration does not uninstall the previous version. To run an earlier release and the new release at the same time is coexistence. To support coexistence, you need to provide non-default port assignments during profile creation.

See the *Getting Started* PDF for more information.

See “Migration and coexistence overview” for more information on coexistence.

Use the following procedure to migrate applications and configurations:

1. Prepare to migrate or update product prerequisites and corequisites to supported versions.
Refer to the IBM WebSphere Application Server supported hardware and software site for current requirements.
2. Install the V6 product.
See the *Getting Started* PDF for more information.
3. Use the Migration wizard to migrate your configuration.
The Migration wizard is new for Version 6; it provides a graphical user interface to the migration tools.
4. Set up multiple versions of WebSphere Application Server to coexist. No run-time conflicts can exist for multiple instances and versions of WebSphere Application Server to run at the same time on the same machine. Potential conflicts can occur with your port assignments. See “Port number settings in WebSphere Application Server versions” on page 107 for more information.
 - a. Run V4 and V6 together, as described in “Setting up Version 4.0.x and Version 6 coexistence” on page 103.
 - b. Run V5 and V6 together, as described in “Setting up Version 5 and Version 6 coexistence” on page 104.
 - c. Run V6 and V6 together, as described in “Setting up Version 6 coexistence” on page 105.
 - d. Create more than one V6 profile on the same machine.
See the *Getting Started* PDF for more information.

You can coexist with, or migrate the applications and configuration from a previous version of WebSphere Application Server.

Migration from V5 to V6 does not require extensive tuning. See Chapter 5, “Migrating product configurations,” on page 65 for a description of fine tuning a migration. Part of the procedure for using the migration tools includes a description of what to tune after using the tools.

For more information on coexistence among releases, see “Migration and coexistence overview.”

Migration and coexistence overview

WebSphere Application Server contains migration tools that provide migration functionality. The Migration wizard is new for Version 6; it provides a graphical user interface to the migration tools. The Migration wizard can call the migration tools, or you can execute them directly. The migration tools migrate applications and configuration information to the new version, as described in Chapter 5, “Migrating product configurations,” on page 65 and Configuration mapping during migration.

Table 1. Overview of migrating from release to release

Migration path	Description
V5.x to V6	<p>The migration from V5.x to V6 is fairly routine. Important reference topics for this migration include:</p> <ul style="list-style-type: none"> • Profile creation, using either the Profile creation wizard or the wasprofile command. See the <i>Getting Started</i> PDF for more information. • Migrating from Version 5 embedded messaging. See the <i>WebSphere Messaging Guide</i> PDF. • Managing WebSphere Application Server, Version 5 JMS use of WebSphere Version 6 messaging resources. See the <i>WebSphere Messaging Guide</i> PDF.
V4.0.x to V6	<p>The migration tools perform a fairly routine migration from V4 to V6. For example, Java 2 Platform, Enterprise Edition (J2EE) 1.2 enterprise archive (EAR) files in V4 work in V6 of WebSphere Application Server, which supports the J2EE 1.4 specification. Similarly, it is not necessary to redeploy enterprise Java bean (EJB) 1.1 Java archive (JAR) files when moving them from V4 to V6, which also supports EJB 2.0 JAR files.</p>

If you neither migrate nor coexist with an earlier version of WebSphere Application Server, you are choosing to ignore the previous installation, and you can run only one version at a time because of conflicting default port assignments. It is possible for both versions to run at the same time without conflict if you use non-default ports in the earlier version. To setup coexistence, select the radio button that states, "Install a new copy of the V6 Application Server product" during installation.

See the *Getting Started* PDF for more information.

You can resolve conflicting port assignments by specifying port assignments for coexistence during profile creation, by wsadmin scripting, or by using the **Servers > Application Servers > server1 > End Points** administrative console page to ensure that Version 6 can run with an earlier version.

Migrating and coexisting have roughly opposite goals. The goal of migration is to reconstruct your earlier version in a nearly identical V6 environment, including applications, configuration settings, universal resource identifier (URI) descriptors, and Web server context roots. The Migration wizard uses the migration tools to migrate the previous configuration and applications. The goal of coexistence is to create an environment that is not in conflict with an earlier version, so far as port settings are concerned. This allows both nodes to start and run at the same time. Coexistence processing changes the following configuration files:

- The `virtualhosts.xml` file:
 - HTTP Transport Port
 - IBM HTTP Server Port
 - HTTPS Transport Port
 - HTTP Administrative Console Port
 - HTTPS Administrative Console Secure Port
- The `serverindex.xml` file:
 - Bootstrap Address
 - SOAP Connector Address
 - DRS Client Address
 - SAS SSL ServerAuth Listener Address
 - CSIV2 SSL ServerAuth Listener Address
 - CSIV2 SSL MutualAuth Listener Address
 - WC adminhost
 - WC defaulthost
 - DCS Unicast Address
 - WC adminhost secure
 - WC default secure
 - SIB Endpoint Address

- SIB Endpoint Secure Address
- SIB MQ Endpoint Address
- SIB MQ Endpoint Secure Address

See “Port number settings in WebSphere Application Server versions” on page 107 for more information.

Consider these issues in a migration or coexistence scenario:

- Conflicting context roots when attempting to share the same Web server.
Follow the procedure in Migrating Web server configurations to learn how to configure a Web server for sharing between WebSphere Application Server versions.
- If your node agent or Application Server has been configured to run as non-root, follow the instructions in “Migrating a previously non-root configuration to root” on page 153 to change the ownership and file permissions of the node directories after running WASPostUpgrade. This must be done before starting the V6 node agent or application server.

Configuration mapping during migration

This topic describes what changes during migration, which always involves migrating a single instance to another single instance on the same machine or a separate machine. Open the information center for the Network Deployment product to learn how the migration tools map models, clones, server groups, clusters, and Lightweight Third Party Authentication (LTPA) security settings.

Many migration scenarios are possible. The Version 6 migration tools maps objects and attributes to the Version 6 environment when you restore a configuration from a previous version.

Bootstrap port

Migration maps a default bootstrap NameServer port setting, 900, from V4.0.x Advanced Edition or V5.x to the V6 NameServer default, 2809. The migration tools map a non-default value directly into the V6 environment.

For V4.0.x Advanced Single Server Edition migration, the bootstrap NameServer port maps to the NameServer value of the Application Server defined in the server configuration file.

Command line parameters

The migration tools convert appropriate command line parameters to Java virtual machine (JVM) settings in the server process definition. Most settings are mapped directly. Some settings, such as memory heap sizes, are not migrated because their roles in the V6 configuration either do not exist, have different meanings, or have different scopes.

Default Server

The name of the default server in Version 6 is server1. All objects previously owned by the DefaultServer of V4.0.x are owned by server1 of Version 6 after migration.

GenericServer

Introduced in V5.1.x, a GenericServer was an APPLICATION_SERVER fitted to manage external resources. In V6, it has its own type, called GENERIC_SERVER. Migration will perform this conversion, but migration cannot accurately migrate the external resources that the GenericServer references. After migration has completed migrating the GenericServer settings, you need to perform the following actions:

- If the old resource that the GenericServer was managing is located under the old WebSphere Application Server installation, you have to copy any related files to the new installation of WebSphere Application Server. You must also run any required setup to put the external application back into a valid and working state. IBM recommends instead that you re-install the resource into the new WebSphere Application Server directory. Whichever you choose to do, the final step is that you need to reset the reference to the new location of the application.
- If the old resource that the GenericServer was managing is not installed under the old WebSphere Application Server installation, nothing further is required.

Java DataBase Connectivity (JDBC) drivers and data sources

Version 6 significantly redefines JDBC and data source objects. The migration tools map V4.0.x data sources to Version 6 data sources, using predecessor settings as input variables.

jmsserver

jmsserver is changed from type MESSAGE_BROKER to type APPLICATION_SERVER. Any queues or topics that it owned have been migrated into the new *WebSphere Platform messaging* framework. In previous releases, in an ND environment, jmsserver was its own MESSAGE_BROKER server; in the Base environment it was contained within APPLICATION_SERVER types. All JMS resources are left untouched and should work without modification. Further migration of these resources can be performed by executing scripts/bats provided by *WebSphere Platform messaging*.

Name bindings

Version 6 has a new naming structure. All references, such as Enterprise JavaBeans (EJB) references that were valid in previous versions no longer work in Version 6. However, you can use the administrative console to add a name binding that maps an old name into the new Version 6 naming structure. For example, the name of the Version 4.0.x enterprise bean reference can be both the name of the binding and the Java Naming and Directory Interface (JNDI) name in the Version 6 name space.

Node name

A Version 4.0.x repository can contain more than one node name and associated children. The WASPostUpgrade tool processes only those objects and children that match the node name of the migrating node. The tool identifies node names in the configuration files that it is migrating and selects any node names in a configuration file that match the long network name or short network name of the migrating machine.

PageList servlet

The configuration of the PageList servlet has changed from V4.0.x. Direct use of the servlet has been deprecated. The PageList servlet is available as part of the servlet extension configuration in the Web archive (WAR) file. All references are updated to the servlet configuration supported in Version 6.

You can also use the Application Server Toolkit (AST) and Rational Web Developer to modify the servlet configuration.

If you use or extend the PageList servlet, you might see an error similar to the following example when running a migrated application that uses the servlet:

```
Error 500: No PageList information is configured for servlet  
EmpInfoApp.SearchByDept
```

Use the Assembly Toolkit to correct the error, by moving the usage or extension to your migrated Enterprise archive (EAR) file:

1. Start the Assembly Toolkit to load the EAR file that generates the error.
2. Open the Web modules within the EAR file.
3. Expand the Web module that generates the error.
4. Open the Web components and find the one that generates the error.
5. Expand the Servlets. The **PageList Extensions** option displays.
6. Add your extension information.
7. Save the EAR file and redeploy it.

Policy file migration from Version 5.x to Version 6

WebSphere Application Server V6 migrates all the policy files that are installed with V5.x by merging settings into the V6 policy files with the following characteristics:

- Any comments located in the V6 policy file will be lost. They are replaced with the comments contained in the V5 policy file.
- Migration will NOT attempt to merge permissions or grants; it is strictly an add-type migration. If the permission or grant is not located in the V6 file, the migration will bring it over.

- Security is a critical component; thus, the migration makes any additions at the end of the original .policy file right after the comment MIGR0372I: Migrated grant permissions follow. This is done to help administrators verify any policy file changes that the migration has made.

Properties directory and the lib/app directory

Migration copies files from prior version directories into the Version 6 configuration. See the following section for more information.

Property file migration from Version 4.0.x

V6 migration migrates property files from V4.0.x if these files are also present in WebSphere Application Server, V6. Specifically, property-file migration includes these files:

- converter.properties
- sas.client.props
- encoding.properties (If the "ko" setting is incorrect, no migration occurs.)
- TraceSettings.properties

You must manually convert settings in other property files to the equivalent V6 configuration.

Property file migration from Version 5.x to Version 6

WebSphere Application Server V6 migrates all the property files that are installed with V5.x by merging settings into the V6 property files with these exceptions:

- j2c.properties (migrated into resources.xml files)
- samples.properties

Migration does not overlay property files.

Resource Adapter Archive (RAR) referenced by J2C resources

RARs that are referenced by J2C resources are migrated if those RARs are in the old WebSphere Application Server installation. In this case, the RARs are copied over to the corresponding location in the new WebSphere Application Server installation. Relational Resource Adapter RARs will not be migrated.

Samples

No migration of Samples from previous versions is available. Equivalent Version 6 Samples are available to use.

Security

Java 2 Security is enabled by default in Version 6. Security enablement might cause some applications to run on Versions 4.0.x and not run on Version 6. Several techniques are available that you can use to define different levels of Java 2 Security in Version 6. One is to create a was.policy file as part of the application, to enable all security permissions. The migration tools call the **wsadmin** command to add an existing was.policy file in the Version 6 properties directory to enterprise applications as they migrate. The migration tools perform this task while moving Version 4.0.x applications into Version 6.

Version 4.0.x introduced properties to support tuning the JNDI search timeout value along with LDAP reuse connection. These two properties are now settings in the Security Center of the Version 6 administrative console. Version 4.0.x property values are not migrated to Version 6 settings.

- The jndi.LDAP.SearchControl.TimeLimit property is equivalent to the Version 6 Search Timeout setting, which is 300 by default in Version 6.
- The jndi.LDAP.URLContextImplementation property is equivalent to the Version 6 Reuse Connection setting, which is true by default in Version 6.

Use the Version 6 administrative console to change these settings to match your Version 4.0.x property values, if necessary.

Servlet package name changes

The package that contains the DefaultErrorReporter, SimpleFileServlet, and InvokerServlet servlets changed in Version 5. In Version 4.0.x, the servlets are in the com.ibm.servlet.engine.webapp class. In Version 6, the servlets are in the com.ibm.ws.webcontainer.servlet class.

Stdin, stdout, stderr, passivation, and working directories

The location for these directories is typically within the installation directory of a previous version. The default location for `stdin`, `stdout`, and `stderr` is the `logs` directory of the Version 6 installation root. The migration tools attempt to migrate existing passivation and working directories. Otherwise, appropriate Version 6 defaults are used.

Using common directories between versions in a coexistence scenario can cause problems.

Transport ports

The migration tools migrate all ports. The tools warn about port conflicts in a log when a port already exists. You must resolve port conflicts before running the servers that are in conflict, at the same time.

Choosing `-scriptCompatibility="true"` or `-scriptCompatibility="false"` results in two different outcomes for transport ports:

- `-scriptCompatibility="true"`: This results in your transport ports being brought over as they are (this is the default).
- `-scriptCompatibility="false"`: This results in the transport ports being converted to the new implementation of channels and chains. From an external application usage standpoint, they will still act the same, but they have been moved to the `TransportChannelService`.

Web modules

The specification level of the Java 2 Platform, Enterprise Edition (J2EE) that Version 6 implements requires behavior changes in the Web container for setting the content type. If a default servlet writer does not set the content type, not only does the Version 6 Web container no longer default to it, the Web container returns the call as `"null"`. This situation might cause some browsers to display resulting Web container tags incorrectly. Migration sets the `autoResponseEncoding` IBM extension to `true` for Web modules as it migrates enterprise applications. This action prevents the problem.

Version 4.0.x to Version 6 migration

The V4.0.x configuration is already at the J2EE 1.2 specification level. Although Version 6 is at the J2EE 1.4 specification level, J2EE 1.2 objects are supported.

- **Enterprise beans**

No redeployment is required when moving EJB 1.1 JAR files from Version 4.0.

Specify only one backend data store vendor per JAR file. If enterprise beans use different backend data stores, package them into separate JAR files.

- **JMS Resources**

All JMS resources from Version 4.0 are mapped into generic JMS resources in the Version 6 configuration. Reconfigure JMS resources that use IBM WebSphere MQ as IBM WebSphere MQ-specific resources. MQ JMS resources have better integration with system management. You do not need to manually define entries in the name space. You can see the backing MQ queue definitions through MQ JMS entries.

- **JSP precompiling**

In Version 4.0.x, the classes generated from JSP pages are in a package based on the directory structure of the WAR file. Any JSP at the top of the context root is in the unnamed package. JSP pages in subdirectories of the root are in packages named after the subdirectories. In Version 6, the classes generated from JSP pages are all in the `org.apache.jsp` package. Therefore, the class files are not compatible between versions.

When migrating an enterprise application from Version 4.0.x to Version 6, recompile the JSP pages to regenerate the class files into the correct packages.

The migration tools provide this support, by using the `-preCompileJSPs` option of the `wsadmin` tool during the installation of the application.

Use the same option to install any Version 4.0.x enterprise applications that you manually move to Version 6.

- **J2EE security**

You can apply security in two Version 4.0.x locations to enterprise applications. Information in the repository has precedence over information in the enterprise application bindings. The migration tools migrate information in the repository to the enterprise application.

- **Secure Sockets Layer (SSL) migration**

The following SSLConfig attributes that point to user-defined key files are migrated from WebSphere Application Server Advanced Edition, Version 4.0.x to V6 as follows:

V4.0.x settings

```
<key_file_name>dir_name/WASLDAPKeyring.jks</key_file_name>
<trust_file_name>dir_name/WASLDAPKeyring.jks</trust_file_name>
```

The *dir_name* variable identifies the original location of the WASLDAPKeyring.jks file.

V6 settings

```
keyFileName="dir_name/WASLDAPKeyring.jks"
trustFileName="dir_name/WASLDAPKeyring.jks"
```

The *dir_name* variable identifies the original location of the WASLDAPKeyring.jks file.

V4.0.x settings

```
<key_file_name>${WAS_HOME}/keys/WASLDAPKeyring.jks</key_file_name>
<trust_file_name>${WAS_HOME}/keys/WASLDAPKeyring.jks</trust_file_name>
```

V6 settings

```
keyFileName="${USER_INSTALL_ROOT}/keys/WASLDAPKeyring.jks"
trustFileName="${USER_INSTALL_ROOT}/keys/WASLDAPKeyring.jks"
```

The migration tools do not copy the key files (for example, .jks, or .kdb) to the corresponding directory in the base WebSphere Application Server product or the Network Deployment product. You must complete the migration of the SSL configuration by copying qualifying key store files to Version 6 directories.

If the key-file-name and trust-file-name attributes point to the DummyServerKeyFile.jks file in the WebSphere Application Server Advanced Edition V4.0.x configuration, the key-file-name and trust-file-name attributes are not migrated to V6. Instead the V6 default value of `${USER_INSTALL_ROOT}/etc/DummyServerKeyFile.jks` is left unchanged.

- **Servlet package name changes when migrating from Version 4.0 to Version 6**

If the web.xml Web module file for Version 4.0 defines the SimpleFileServlet servlet, the migration tools update the class name to reflect the Version 6 package. The tools also set the FileServing Enabled attribute to true.

If the web.xml file defines the InvokerServlet servlet, the migration tools update the class name to reflect the Version 6 package. The tools also set the ServeServletsByClassnameEnabled attribute to true.

If the web.xml file defines the DefaultErrorReporter servlet, the migration tools update the class name to reflect the Version 6 package.

Version 5.x to Version 6 migration

Migrating from V5.x to V6 is much less complicated than migrating from V4.0.x. Both versions use the same underlying definitions. The task involves mapping configuration files from the V5.x to the V6 configuration and copying installed applications into the new product. The migration tools support the migration of federated nodes and support the full migration of a Network Deployment node.

Java heap size for migrating EAR files

When migrating all 5.x EAR files to V6 using the wsadmin tool, the WASPostUpgrade tool uses the default maximum Java heap size value of 64MB to install the EAR files.

If a V5.x EAR file fails to install during migration because the Java heap size is not large enough, you see a message similar to the following error:

```
java.lang.OutOfMemoryError JVMXE006:OutOfMemoryError
```

Increase the maximum Java heap size and follow the instructions below to install the application:

Installing the application on WebSphere Application Server, Version 6

Assume that:

Installation root

C:\WebSphere\AppServer

Number signs (###)

Maximum heap size value

EAR_file_name

The name of the EAR file

app_name

The name of the application.

server_name

The name of the server on which the EAR file installs

node_name

The name of the node on which the server is configured

The command appears on more than one line for clarity.

```
wsadmin -conntype NONE
        -javaoption
        -Xmx###m
        -c "$AdminApp install
           C:\\WebSphere\\AppServer\\installableApps\\
           EAR_file_name
        {-nodeployejb
        -appname app_name
        -server server_name
        -node node_name}"
```

Installing the application on WebSphere Application Server Network Deployment, Version 6

Assume that:

Installation root

C:\WebSphere\DeploymentManager

Number signs (###)

Maximum heap size value

EAR_file_name

The name of the EAR file

app_name

The name of the application.

cluster_name

The name of the cluster on which the EAR file should be installed

The command appears on more than one line for clarity.

```
wsadmin -conntype NONE
        -javaoption
        -Xmx###m
        -c "$AdminApp install
           C:\\WebSphere\\DeploymentManager\\installableApps\\
           EAR_file_name>
        {-nodeployejb
        -appname app_name
        -cluster cluster_name}"
```

Migrating WebSphere programming model extensions (PMEs)

This topic describes the movement of a subset of PMEs to WebSphere Application Server Version 6 from WebSphere Application Server Enterprise Edition V4.0.x and V5.0.x and WebSphere Business Integration Server Foundation V5.1.x.

Overview of PME migration

The migration of PME services from WebSphere Application Server Enterprise Edition V4.0.x and V5.0.x and WebSphere Business Integration Server Foundation V5.1.x to WebSphere Application Server Version 6 is handled on an individual basis. For PME services that are not supported in WebSphere Application Server V6, all configuration information is removed. For PME services that are supported in WebSphere Application Server V6, the configuration from the previous release environment overwrites the values in the new release.

Validating PMEs

As part of application installation, both during migration and outside of migration, applications are validated to ensure that they only use resources for services that are supported by WebSphere Application Server V6. Any application that uses a resource for a service that is not supported by WebSphere Application Server V6 will not work correctly, and an error will be issued indicating that the application cannot be installed.

PME-specific information

For more information on the specific PMEs that have been migrated to WebSphere Application Server V6, see the *Developing and Deploying Applications* PDF.

Part 2. Migrating

Chapter 4. Migrating applications

Use this section to learn about migrating applications.

Web applications

Migrating Web application components from WebSphere Application Server Version 4.x

Migration of Web applications deployed in WebSphere Application Server Version 5.x is not necessary; version 2.2 and 2.3 of the Java Servlet specification and version 1.2 and 1.4 of the JavaServer Pages (JSP) specification are still supported. However, where there are behavioral differences between the Java 2 Enterprise Edition (J2EE) 1.2 and J2EE 1.3 specifications, bear in mind that J2EE 1.3 specifications are implemented in WebSphere Application Server Version 5 and will override any J2EE 1.2 behaviors.

Servlet migration might be a concern if your application:

- Implements a WebSphere internal servlet to bypass a WebSphere Application Server Version 4.x single application path restriction.
- Extends a PageListServlet that relies on configuration information in the servlet configuration XML file.
- Uses a servlet to generate Hyper Text Markup Language (HTML) output.
- Calls the `response.sendRedirect` method for a servlet using the `encodeRedirectURL` function or executing within a non-context root.

JSP migration might be a concern if your application references JSP page implementation classes in unnamed packages, or if you install WebSphere Application Server Version 4.x EAR files (deployed in Version 4.x with the JSP Precompile option), in Version 5.

Follow these steps if migration issues apply to your Web application:

1. Use WebSphere Application Server Version 5 package names for any WebSphere Application Server Version 4.x internal servlets, which are implemented in your application.

In WebSphere Application Server Version 4.x, Web modules with a context root setting of / are not supported. Accessing Web modules with this root context results in HTTP 404 - File not Found errors.

To bypass the errors, and to enable the serving of static files from the root context, WebSphere Application Server Version 4.x users are advised to add the servlet class, `com.ibm.servlet.engine.webapp.SimpleFileServlet`, to their Web module.

The Version 4.x single path limitation does not exist in Version 5. However, users who choose to use the `com.ibm.servlet.engine.webapp.SimpleFileServlet` in Version 5 must do one of the following:

- Rename `com.ibm.servlet.engine.webapp.SimpleFileServlet` to `com.ibm.ws.webcontainer.servlet.SimpleFileServlet`.
- Open a Web deployment descriptor editor using an assembly tool and select **File serving enabled** on the **Extensions** tab.

The following list identifies the other internal servlets affected by the Version 6 package name change:

- DefaultErrorReporter
- AutoInvoker

Use the Version 5 package name, `com.ibm.ws.webcontainer.servlet.servlet class name` for these servlets.

2. Use the WASPostUpgrade tool to migrate servlets that extend PageListServlet and rely on configuration information in the associated XML servlet configuration file. In Version 4.x, the XML servlet configuration file provides configuration data for page lists and augments servlet configuration information. This file is named as either `servlet_class_name.servlet` or `servlet_name.servlet`, and is stored in the same directory as the servlet class file.

The XML servlet configuration file is not supported in WebSphere Application Server Version 5.

3. Set a content type if your servlet generates Hyper Text Markup Language (HTML) output.
The default behavior of the Web container changed in WebSphere Application Server Version 5. If the servlet developer does not specify a content type in the servlet then the container is forbidden to set one automatically. Without an explicit content type setting, the content type is set to null. The Netscape browser displays HTML source as plain text with a null content type setting.

To resolve this problem, do one of the following:

- Explicitly set a content type in your servlet.
- Open a Web deployment descriptor editor in an assembly tool and select **Automatic Response Encoding enabled** on the **Extensions** tab.

4. Set the Java environment variable, `com.ibm.websphere.sendredirect.compatibility`, to **true** if you want your URLs interpreted relative to the application root.

The default value of the Java environment variable `com.ibm.websphere.sendredirect.compatibility` changed in WebSphere Application Server Version 5. In Version 4, the default setting of this variable is true. In Version 5, the setting is false.

When this variable is set to false, if a URL has a leading slash, the URL is interpreted relative to the Web module/application root. However, if the URL does not have a leading slash, it is interpreted relative to the Web container root (also known as the Web server document root). Therefore, if an application has a WAR file that has a context root of `myPledge_app` and a servlet that has a servlet mapping of `/Intranet/`, a JSP file in the WAR file cannot access the servlet when its `encodeRedirectURL` is set to `/Intranet/myPledge`. The JSP file can access the servlet if the `encodeRedirectURL` is set to `myPledge_app/Intranet/myPledge`, or if the `com.ibm.websphere.sendredirect.compatibility` variable is set to true.

5. Use the WASPostUpgrade tool to migrate WebSphere Version 4.x enterprise applications to Version 5.

Note: The WebSphere Application Server Version 4.x JSP page implementation class files are not compatible with the WebSphere Application Server Version 5 JSP container.

The WASPostUpgrade tool automatically precompiles JSP files, which ensures the JSP page implementation class files are compatible with Version 5.

If you install Version 4.x EAR files, deployed with the JSP Precompile option, in Version 5, and you choose not to follow the migration path, do one of the following:

- Select the Pre-compile JSP option in the administrative console Install New Application window.
- Specify the `-preCompileJSPs` option when using the `wsadmin` tool.

6. Import your classes if your application uses unnamed packages.

Section 8.2 of the JSP 1.2 specification states:

The JSP container creates a JSP page implementation class for each JSP page. The name of the JSP page implementation class is implementation dependent. The JSP page implementation object belongs to an implementation-dependent named package. The package used may vary between one JSP and another, so minimal assumptions should be made. The unnamed package should not be used without an explicit *import* of the class.

For example, if `myBeanClass` is in the unnamed package, and you reference it in a `jsp:useBean` tag, then you must explicitly import `myBeanClass` with the page directive `import` attribute, as shown in the following example:

```
<%@page import="myBeanClass" %>

<jsp:useBean id="myBean" class="myBeanClass" scope="session"/>
```

In WebSphere Application Server Version 5, the JSP engine creates JSP page implementation classes in the `org.apache.jsp` package. If a class in the unnamed package is not explicitly imported, then the `javac` compiler assumes the class is in package `org.apache.jsp`, and the compilation fails. **Note:** Avoid using the unnamed package altogether because of a change made in JDK 1.4 that will affect the JSP 2.0 specification. WebSphere Application Server Version 5 ships with JDK 1.3.1, so this is not an issue with the Version 5 JSP engine, but it will become an issue in future releases.

The *Incompatibilities* section of the version 1.4 Java 2 Platform, Standard Edition (J2SE) documentation states:

The compiler now rejects import statements that import a type from the unnamed namespace. Previous versions of the compiler would accept such import declarations, even though they were arguably not allowed by the language (because the type name appearing in the import clause is not in scope). The specification is being clarified to state clearly that you cannot have a simple name in an import statement, nor can you import from the unnamed namespace.

To summarize, the syntax:

```
import SimpleName;
```

is no longer legal. Nor is the syntax

```
import ClassInUnnamedNamespace.Nested;
```

which would import a nested class from the unnamed namespace.

To fix such problems in your code, move all of the classes from the unnamed namespace into a named namespace.

See "Resources for learning" for links to the J2SE, JSP, and Servlet specification documentation.

Migrating Web application components from WebSphere Application Server Version 5.x

Supported open specification levels in WebSphere Application Server Version 6 are documented in article, [Migrating](#).

Migration of Web applications deployed in WebSphere Application Server Version 5.x is not necessary; version 2.2 and 2.3 of the Java Servlet specification and version 1.2 and 1.4 of the JavaServer Pages (JSP) specification are still supported. However, where there are behavioral differences between the Java 2 Enterprise Edition (J2EE) 1.2 and J2EE 1.3 specifications, bear in mind that J2EE 1.3 specifications are implemented in WebSphere Application Server Version 5 and will override any J2EE 1.2 behaviors.

Servlet migration might be a concern if your application:

- implements a WebSphere internal servlet to bypass a WebSphere Application Server Version 4.x single application path restriction.
- extends a `PageListServlet` that relies on configuration information in the servlet configuration XML file.
- uses a servlet to generate Hyper Text Markup Language (HTML) output.
- calls the `response.sendRedirect` method for a servlet using the `encodeRedirectURL` function or executing within a non-context root.

JSP migration might be a concern if your application references JSP page implementation classes in unnamed packages, or if you install WebSphere Application Server Version 4.x EAR files (deployed in Version 4.x with the `JSP Precompile` option), in Version 5. You need to recompile all JSP pages when migrating from WebSphere Application Server Version 5.x to version 6.

Follow these steps if migration issues apply to your Web application:

Use WebSphere Application Server Version 5 package names for any WebSphere Application Server Version 4.x internal servlets, which are implemented in your application.

In WebSphere Application Server Version 4.x, Web modules with a context root setting of `/` are not supported. Accessing Web modules with this root context results in HTTP 404 - File not Found errors.

To bypass the errors, and to enable the serving of static files from the root context, WebSphere Application Server Version 4.x users are advised to add the servlet class, `com.ibm.servlet.engine.webapp.SimpleFileServlet`, to their Web module.

The Version 4.x single path limitation does not exist in Version 5. However, users who choose to use the `com.ibm.servlet.engine.webapp.SimpleFileServlet` in Version 5 must do one of the following:

- Rename `com.ibm.servlet.engine.webapp.SimpleFileServlet` to `com.ibm.ws.webcontainer.servlet.SimpleFileServlet`.
- Open a Web deployment descriptor editor using an assembly tool and select **File serving enabled** on the **Extensions** tab.

See "Resources for learning" for links to the J2SE, JSP, and Servlet specification documentation.

Migrating HTTP sessions

Note: In Version 5 and higher, default write frequency mode is `TIME_BASED_WRITES`, which is different from Version 4.0.x default mode of `END_OF_SERVICE`.

Migrating from Version 5.x

No programmatic changes are required to migrate from version 5.x to version 6.

Migrating from Version 4.0

No programmatic changes are required to migrate from version 4.0.x to version 6.

Migrating enterprise bean code to the supported specification

Support for Version 2.1 of the Enterprise JavaBeans (EJB) specification is added for Version 6 of this product. Migration of enterprise beans deployed in Versions 4 or 5 of this product is not generally necessary; Versions 1.1 and 2.0 of the EJB specification are still supported. Follow these steps as appropriate for your application deployment.

1. Modify enterprise bean code for changes in the specification.
 - For Version 1.0 beans, migrate at least to Version 1.1.
See the *WebSphere Messaging Guide* PDF for more information.
 - As stated previously, migration from Version 1.1 to Version 2.x of the EJB specification is not required for redeployment on this version of the product. However, if your application requires the capabilities of Version 2.x, migrate your Version 1.1-compliant code.
See the *WebSphere Messaging* PDF for more information.

Note: The EJB Version 2.0 specification mandates that prior to the EJB container's running a `findByMethod` query, the state of all enterprise beans enlisted in the current transaction be synchronized with the persistent store. (This is so the query is performed against current data.) If Version 1.1 beans are reassembled into an EJB 2.x-compliant module, the EJB container synchronizes the state of Version 1.1 beans as well as that of Version 2.x beans. As a result, you might notice some change in application behavior even though the application code for the Version 1.1 beans has not been changed.

2. You might have to modify code for some EJB 1.1-compliant modules that were not migrated to Version 2.x. Use the following information to help you decide.
 - Some stub classes for deployed enterprise beans have changed in the Java 2 Software Development Kit, Version 1.4.1.
 - The task of generating deployment code for enterprise beans changed significantly for EJB 1.1-compliant modules relative to EJB 1.0-compliant modules.

For detailed information about source and binary compatibility between deployed versions, see "Resources for learning."

3. Reassemble and redeploy all modules to incorporate migrated code.

Migrating enterprise bean code from Version 1.0 to Version 1.1

The following information generally applies to any enterprise bean that currently complies with Version 1.0 of the Enterprise JavaBeans (EJB) specification. For more information about migrating code for beans produced with Rational Application Developer, see the documentation for that product.

1. In session beans, replace all uses of `javax.jts.UserTransaction` with `javax.transaction.UserTransaction`. Entity beans may no longer use the `UserTransaction` interface at all.
2. In finder methods for entity beans, include `FinderException` in the `throws` clause.
3. Remove `throws` of `java.rmi.RemoteException`; throw `javax.ejb.EJBException` instead. However, continue to include `RemoteException` in the `throws` clause of `home` and `remote` interfaces as required by the use of `Remote Method Invocation (RMI)`.
4. Remove uses of the `finalize()` method.
5. Replace calls to `getCallerIdentity()` with calls to `getCallerPrincipal()`. The use of `getCallerIdentity()` is deprecated.
6. Replace calls to `isCallerInRole(Identity)` with calls to `isCallerInRole (String)`. The use of `isCallerInRole(Identity)` and `java.security.Identity` is deprecated.
7. Replace calls to `getEnvironment()` in favor of JNDI lookup. As an example, change the following code:

```
String homeName =
    aLink.getEntityContext().getEnvironment().getProperty("TARGET_HOME_NAME");
if (homeName == null) homeName = "TARGET_HOME_NAME";
```

The updated code would look something like the following:

```
Context env = (Context)(new InitialContext()).lookup("java:comp/env");
String homeName = (String)env.lookup("ejb10-properties/TARGET_HOME_NAME");
```

8. In `ejbCreate` methods for an entity bean with container-managed persistence (CMP), return the bean's primary key class instead of `void`.
9. Add the `getHomeHandle()` method to `home` interfaces.

```
public javax.ejb.HomeHandle getHomeHandle() {return null;}
```

Consider enhancements to match the following changes in the specification:

- Primary keys for entity beans can be of type `java.lang.String`.
- Finder methods for entity beans return `java.util.Collection`.
- Check the format of any JNDI names being used. Local name spaces are also supported.
- Security is defined by role, and isolation levels are defined at the method level rather than at the bean level.

Migrating enterprise bean code from Version 1.1 to Version 2.1

Enterprise JavaBeans (EJB) Version 2.1-compliant beans can be assembled only in an EJB 2.1-compliant module, although an EJB 2.1-compliant module can contain a mixture of Version 1.x and Version 2.1 beans.

The EJB Version 2.1 specification mandates that prior to the EJB container starting a `findByMethod` query, the state of all enterprise beans that are enlisted in the current transaction be synchronized with the persistent store. (This action is so the query is performed against current data.) If Version 1.1 beans are reassembled into an EJB 2.1-compliant module, the EJB container synchronizes the state of Version 1.1 beans as well as that of Version 2.1 beans. As a result, you might notice some change in application behavior even though the application code for the Version 1.1 beans has not been changed.

The following information generally applies to any enterprise bean that currently complies with Version 1.1 of the EJB specification. For more information about migrating code for beans produced with the Rational Application Developer tool, see the documentation for that product.

1. In beans with container-managed persistence (CMP) version 1.x, replace each CMP field with abstract `get` and `set` methods. In doing so, you must make each bean class abstract.

2. In beans with CMP version 1.x, change all occurrences of `this.field = value` to `setField(value)`.
3. In each CMP bean, create abstract get and set methods for the primary key.
4. In beans with CMP version 1.x, create an EJB Query Language statement for each finder method.

Note: EJB Query Language has the following limitations in Application Developer Version 5:

- EJB Query Language queries involving beans with keys made up of relationships to other beans appear as invalid and cause errors at deployment time.
 - The IBM EJB Query Language support extends the EJB 2.1 spec in various ways, including relaxing some restrictions, adding support for more DB2 functions, and so on. If portability across various vendor databases or EJB deployment tools is a concern, then care should be taken to write all EJB Query Language queries strictly according to instructions described in Chapter 11 of the EJB 2.1 specification.
5. In finder methods for beans with CMP version 1.x, return `java.util.Collection` instead of `java.util.Enumeration`.
 6. Update handling of non-application exceptions.
 - To report non-application exceptions, throw `javax.ejb.EJBException` instead of `java.rmi.RemoteException`.
 - Modify rollback behavior as needed: In EJB versions 1.1 and 2.1, all non-application exceptions thrown by the bean instance result in the rollback of the transaction in which the instance is running; the instance is discarded. In EJB 1.0, the container does not roll back the transaction or discard the instance if it throws `java.rmi.RemoteException`.
 7. Update rollback behavior as the result of application exceptions.
 - In EJB versions 1.1 and 2.1, an application exception does not cause the EJB container to automatically roll back a transaction.
 - In EJB Version 1.1, the container performs the rollback only if the instance has called `setRollbackOnly()` on its `EJBContext` object.
 - In EJB Version 1.0, the container is required to roll back a transaction when an application exception is passed through a transaction boundary started by the container.
 8. Update any CMP setting of application-specific default values to be inside `ejbCreate` (not using global variables, since EJB 1.1 containers set all fields to generic default values before calling `ejbCreate`, which overwrites any previous application-specific defaults). This approach also works for EJB 1.0 CMPs.

Note: In Application Developer Version 5, there is a J2EE Migration wizard to migrate the EJB beans within an EJB 2.1 project from 1.x into 2.1 (you cannot just migrate individually selected beans). The wizard performs migration steps #1 to #2 above. It also migrates EJB 1.1 (proprietary) relationships into EJB 2.1 (standard) relationships, and maintains EJB inheritance.

Migrating Apache SOAP Web services to Web Services for J2EE standards

This topic explains how to migrate Web services that were developed using Apache SOAP to Web services that are developed based on the Web Services for Java 2 Platform, Enterprise Edition (J2EE) specification.

If you have used Web services based on Apache SOAP and now want to develop and implement Web services based on the Web Services for J2EE specification, you need to migrate client applications developed with all versions of 4.0, and versions of 5.0 prior to 5.0.2.

To migrate these client applications according to the Web Services for J2EE standards:

1. Plan your migration strategy. You can port an Apache SOAP client to a Java API for XML-based RPC (JAX-RPC) Web services client in one of two ways:

- If you have, or can create, a Web Services Description Language (WSDL) document for the service, consider using the **WSDL2Java** command tool to generate bindings for the Web service. It is more work to adapt an Apache SOAP client to use the generated JAX-RPC bindings, but the resulting client code is more robust and easier to maintain. To follow this path, develop a Web services client based on Web Services for J2EE.
- If you do not have a WSDL document for the service, do not expect the service to change, and you want to port the Apache SOAP client with minimal work, you can convert the code to use the JAX-RPC dynamic invocation interface (DII), which is similar to the Apache SOAP APIs. The DII APIs do not use WSDL or generated bindings.

Because JAX-RPC does not specify a framework for user-written serializers, the JAX-RPC does not support the use of custom serializers. If your application cannot conform to the default mapping between Java, WSDL, and XML technology supported by WebSphere Application Server, do not attempt to migrate the application. The remainder of this topic assumes that you decided to use the JAX-RPC dynamic invocation interface (DII) APIs.

2. Review the GetQuote Sample. A Web services migration Sample is available in the Samples Gallery. This Sample is located in the GetQuote.java file, originally written for Apache SOAP users, and includes an explanation about the changes needed to migrate to the JAX-RPC DII interfaces.
3. Convert the client application from Apache SOAP to JAX-RPC DII The Apache SOAP API and JAX-RPC DII API structures are similar. You can instantiate and configure a call object, set up the parameters, invoke the operation, and process the result in both. You can create a generic instance of a Service object with the following command:

```
javax.xml.rpc.Service service = ServiceFactory.newInstance().createService(new QName(""));
```

in JAX-RPC.

- a. Create the Call object. An instance of the Call object is created with the following code:

```
org.apache.soap.rpc.Call call = new org.apache.soap.rpc.Call ()
```

in Apache SOAP.

An instance of the Call object is created by

```
java.xml.rpc.Call call = service.createCall();
```

in JAX-RPC.

- b. Set the endpoint Uniform Resource Identifiers (URI). The target URI for the operation is passed as a parameter to

```
call.invoke: call.invoke("http://...", "");
```

in Apache SOAP.

The setTargetEndpointAddress method is used as a parameter to

```
call.setTargetEndpointAddress("http://...");
```

in JAX-RPC.

Apache SOAP has a setTargetObjectURI method on the Call object that contains routing information for the request. JAX-RPC has no equivalent method. The information in the targetObjectURI is included in the targetEndpoint URI for JAX-RPC.

- c. Set the operation name. The operation name is configured on the Call object by

```
call.setMethodName("opName");
```

in Apache SOAP.

The setOperationName method, which accepts a QName instead of a String parameter, is used in JAX-RPC as illustrated in the following example:

```
call.setOperationName(new javax.xml.namespace.Qname("namespace", "opName"));
```

- d. Set the encoding style. The encoding style is configured on the Call object by

```
call.setEncodingStyleURI(org.apache.soap.Constants.NS_URI_SOAP_ENC);
```

in Apache SOAP.

The encoding style is set by a property of the Call object

```
call.setProperty(javax.xml.rpc.Call.ENCODINGSTYLE_URI_PROPERTY, "http://schemas.xmlsoap.org/soap/encoding/");
```

in JAX-RPC.

- e. Declare the parameters and set the parameter values. Apache SOAP parameter types and values are described by parameter instances, which are collected into a vector and set on the Call object before the call, for example:

```
Vector params = new Vector ();
params.addElement (new org.apache.soap.rpc.Parameter(name, type, value, encodingURI));
// repeat for additional parameters...
call.setParams (params);
```

For JAX-RPC, the Call object is configured with parameter names and types without providing their values, for example:

```
call.addParameter(name, xmlType, mode);
// repeat for additional parameters
call.setReturnType(type);
```

Where

- *name* (type `java.lang.String`) is the name of the parameter
- *xmlType* (type `javax.xml.namespace.QName`) is the XML type of the parameter
- *mode* (type `javax.xml.rpc.ParameterMode`) the mode of the parameter, for example, IN, OUT, or INOUT

- f. Make the call. The operation is invoked on the Call object by

```
org.apache.soap.Response resp = call.invoke(endpointURI, "");
```

in Apache SOAP.

The parameter values are collected into an array and passed to `call.invoke` as illustrated in the following example:

```
Object resp = call.invoke(new Object[] {parm1, parm2,...});
```

in JAX-RPC.

- g. Check for faults. You can check for a SOAP fault on the invocation by checking the response:

```
if resp.generatedFault then {
org.apache.soap.Fault f = resp.getFault();
f.getFaultCode();
f.getFaultString();
}
```

in Apache SOAP.

A `java.rmi.RemoteException` error is displayed in JAX-RPC if a SOAP fault occurs on the invocation.

```
try {
... call.invoke(...)
} catch (java.rmi.RemoteException) ...
```

- h. Retrieve the result. In Apache SOAP, if the invocation is successful and returns a result, it can be retrieved from the Response object:

```
Parameter result = resp.getReturnValue(); return result.getValue();
```

In JAX-RPC, the result of `invoke` is the returned object when no exception is displayed:

```
Object result = call.invoke(...);
...
return result;
```

You have migrated Apache SOAP Web services to J2EE Web services.

Develop a Web services client. This topic explains how to develop a Web services client based on the Web Services for J2EE specification.

Test the Web services-enabled clients to make sure the migration process is successful and you can implement the Web services in a J2EE environment.

Migrating to Version 3 of the UDDI Registry

Use this topic to migrate to a UDDI Registry that uses a DB2 database, from a previous UDDI Version to UDDI Version 3.

Use this topic to migrate to a UDDI Registry that uses a DB2 database, from a Version 2 UDDI Registry to a Version 3 UDDI Registry.

Before you begin.

You can use the process described in this topic to migrate a UDDI Registry to UDDI Version 3, running in WebSphere Application Server Version 6, subject to the following constraints:

- Your registry uses a DB2 database.
- Your registry runs in WebSphere Application Server Version 5 or later.

If you are migrating from the IBM WebSphere UDDI Version 1.1 or 1.1.1, which ran on WebSphere Application Server Version 4, you should first migrate to UDDI Version 2 running on WebSphere Application Server Version 5 (as described in the WebSphere Application Server Version 5 Information Center), then complete the steps described in this topic.

You *should not* run the version 2 UDDI Registry application from WebSphere Application Server Version 5 and onwards except in the case where you are running in a mixed cell migration mode, where WebSphere Application Server version 6 Deployment Manager manages the version 5.x nodes. A Version 2 UDDI Registry can run in WebSphere Application Server v6 but it is *not* supported.

This task describes in detail the process of migrating from a Version 2 UDDI Registry running on WebSphere Application Server Version 5, to a Version 3 UDDI Registry running on WebSphere Application Server Version 6.

If you have UDDI deployed in a clustered application server and you migrate to WebSphere Application Server Version 6, you will not be able to run UDDI across the mixed-version cluster. You can continue to run the UDDI Registry on the server(s) that remain at Version 5 of WebSphere Application Server, but you will not be able to run the UDDI Registry in the Version 6 servers of WebSphere Application Server until all nodes in the cluster have been migrated to WebSphere Application Server Version 6. This is because the UDDI data needs to be migrated from the UDDI Version 2 format to the UDDI Version 3 format.

If you are migrating the UDDI Registry from a WebSphere Application Server version 5.x Network Deployment configuration, or from a WebSphere Application Server version 5.x standalone application server, the steps are very similar. For a Network Deployment migration to a WebSphere Application Server version 6 Network Deployment configuration, a number of choices are available, including having a mixed version cell where the WebSphere Application Server version 6 Deployment Manager can manage application servers at different levels. In this way, individual application servers can be migrated when convenient in a step by step manner. See Migrating and coexisting overview elsewhere in this Information Center for more details.

To migrate a Version 2 UDDI Registry to UDDI Version 3, running in WebSphere Application Server Version 6, complete the following steps (this is assuming that you have migrated the WebSphere Application Manager Version 5 Deployment Manager to Version 6 first):

1. Stop the UDDI Registry application that is running in your Version 5.x application server. This prevents further UDDI requests being directed to the UDDI Registry and ensures that no new data is published during the migration process.
2. Record information about the `uddi.properties` values being used. This file is located in the `<DeploymentManager-install-dir>/config/cells/<cellname>/nodes/<nodename>/servers/<servername>` directory on your WebSphere Application Server Version 5.x system (or in the `properties` subdirectory if you are migrating a standalone application server).
3. Migrate from WebSphere Application Server Version 5.x to Version 6 (see Migrating and coexisting overview). This results in a new directory tree for the migrated Version 6 application server.
4. Start the new migrated (Version 6) application server.
5. Create the UDDI Version 3 DB2 database.
6. Using the administrative console or JMX:
 - a. Create a UDDI Version 3 JDBC provider and datasource to access the UDDI Version 3 database.
 - b. Create a J2C authentication data entry for the Version 3 UDDI database.
 - c. Create a new datasource for the Version 2 UDDI database. This is known as the UDDI Migration datasource. Note that the JNDI name **must** be `datasources/uddimigration`, or that you would like to keep the same (see table below).
7. Run the `uddiDeploy.jacl` script. This deploys the UDDI Version 3 application from the `installableApps` directory into the application server, supplying the node name and server name as parameters.
8. Start the UDDI Application from the administrative console.
9. Configure the node (using the UDDI administrative console or JMX management interface). You must do this before you complete node initialization (see next step). It is recommended you take all the default values except for:
 - The `NodeId` and related 'node personality' settings, which must be set to suitable, and valid, values.
 - Settings from `uddi.properties` that **must** remain the same as were used for Version 2.
 - Settings from `uddi.properties` that you would like to keep the same (such as `dbMaxResultCount`).
 - Any policy values that you wish to change from the default settings (or these can be changed at a later time).
10. Request the completion of node initialization using the Initialize button on the UDDI administrative console (or using the JMX management interface for UDDI). The UDDI Node Initialization detects that there is a UDDI Migration datasource, and migrates the Version 2 data as part of the UDDI node initialization processing.

Note: This step can take some time depending on the amount of data in your UDDI Registry.
11. Verify the migration process has completed successfully. The following message should appear in the server log:


```
CWUDQ0003I: UDDI registry migration has completed
```

 Should the following error appear:


```
CWUDQ0004W: UDDI registry not started due to migration errors
```

 it means that an unexpected error has occurred during migration. The UDDI Registry node will not be activated. Check the error logs for the problem and, if it cannot be fixed contact your IBM representative for advice.
12. Once migration is complete the UDDI Migration datasource may be removed, and the registry is available for use.

Migrating `uddi.properties` values

The table below shows the values of the properties that were stored in the `uddi.properties` file in WebSphere Application Server V5, and their equivalents in WebSphere Application Server v6.

Version 2 UDDI property (set in <code>uddi.properties</code> file)	Version 3 UDDI Property (set via Administrative Console or UDDI Administrative Interface)	Recommended Version 3 UDDI property setting
<code>dbMaxResultCount</code>	maximum inquiry response set size	You might want to retain the value from Version 2, but can safely change this (or use the default)
<code>persist</code>	no equivalent	Not applicable
<code>defaultLanguage</code>	default language code	You are recommended to retain the value from Version 2
<code>operatorName</code>	UDDI node ID	You must use a valid value for the UDDI node ID. This will be applied to your Version 2 data as it is migrated.
<code>maxSearchKeys</code>	maximum search keys	You might want to retain the value from Version 2, but can safely change this (or use the default)
<code>getServletURLprefix</code>	Prefix for generated discoveryURLs	You should enter a valid value for your configuration, which should therefore be the same as the value used for Version 2.
<code>getServletName</code>	no equivalent	Not applicable

Setting up a UDDI Migration Datasource

This topic describes how to set up a UDDI Migration Datasource, to be used to reference a Version 2 UDDI Registry database. Migration is only supported from DB2, so these instructions describe how to set up a DB2 datasource.

1. If a suitable DB2 JDBC Provider does not already exist, then create one, selecting the options DB2 Legacy CLI-based Type 2 JDBC Driver and Connection Pool datasource.
2. Create a datasource for the Version 2 UDDI Registry database:

- a. Expand **Resource** and **JDBC Providers**.
- b. Select the desired 'scope' of the JDBC provider you selected or created earlier. For example, select:

Server: `yourservname`

This displays the JDBC providers at the server level.

- c. Select the JDBC provider created or selected earlier.
- d. Under **Additional Properties**, select **Data Sources** (not the Data Sources Version 4 option).
- e. Select **New** to create a new datasource.
- f. Fill in the details for the datasource as follows:

Name a suitable name, such as UDDI Datasource

JNDI name

set to `datasources/uddimigration` - this value is compulsory, and must be as shown.#

Description

a suitable description

Category

set to `uddi`

Data store helper class name

filled in for you as: com.ibm.websphere.rsadapter.DB2DataStoreHelper

datasource properties

...

DB2 databasename

UDDI20, or the name given to your Version 2 UDDI DB2 database

Component-managed authentication alias

select the alias for the DB2 userid used to access UDDI data, for example MyNode/UDDIAlias (refer to Setting up and deploying a new UDDI Registry if you do not have such an alias)

Container-managed authentication alias

Set to (none)

Mapping-configuration alias

Set to (none)

Leave all the other fields unchanged.

3. Click **Apply** and save the changes to the Master Configuration.

Using a remote database for the UDDI Registry

It is possible for the UDDI Registry database to be hosted on a separate system (remote system) from the system on which the UDDI Registry application is deployed.

This is achieved using standard database capabilities of the database product used for the UDDI Registry database. You should refer to documentation for the database product if you are not familiar with these capabilities. Some considerations specific to each database product are:

Remote DB2

Create a DB2 UDDI database on the remote system, and use the DB2 Client to create an alias to reference it. Use the alias name as the Database name in the UDDI datasource.

Remote Oracle

Create the Oracle tables used for an Oracle UDDI database on the remote system, and use the URL property of the UDDI datasource to reference the remote Oracle instance.

Networked Cloudscape

Create a Cloudscape UDDI database on the remote system, and use the Cloudscape Network Server using Universal data source properties (Database name, Server name and Port number) of the UDDI datasource to reference the remote Cloudscape database.

Migrating a Version 4.0 data access application to Version 6.0

To use the connection management infrastructure in WebSphere Application Server Version 6.0, you must package your application as a J2EE 1.3 (or later) application. This process involves repackaging your Web modules to the 2.3 specification and your EJB modules to the 2.1 specification before installing them onto WebSphere Application Server.

Converting a 2.2 Web module to a 2.3 Web module

Use the following steps to migrate each of your Web modules.

1. Open an assembly tool such as the Application Server Toolkit (AST) or Rational Web Developer.
2. Create a new Web module by selecting **File > New > Web Module**.
3. Add any required class files to the new module.
 - a. Expand the **Files** portion of the tree.
 - b. Right-click **Class Files** and select **Add Files**.
 - c. In the Add Files window, click **Browse**.
 - d. Navigate to your WebSphere Application Server 4.0 EAR file and click **Select**.
 - e. In the upper left pane of the Add Files window, navigate to your WAR file and expand the `WEB-INF` and `classes` directories.
 - f. Select each of the directories and files in the classes directory and click **Add**.
 - g. After you add all of the required class files, click **OK**.
4. Add any required JAR files to the new module.
 - a. Expand the **Files** portion of the tree.
 - b. Right-click **Jar Files** and select **Add Files**.
 - c. Navigate to your WebSphere 4.0 EAR file and click **Select**.
 - d. In the upper left pane of the Add Files window, navigate to your WAR file and expand the `WEB-INF` and `lib` directories.
 - e. Select each JAR file and click **Add**.
 - f. After you add all of the required JAR files, click **OK**.
5. Add any required resource files, such as HTML files, JSP files, GIFs, and so on, to the new module.
 - a. Expand the **Files** portion of the tree.
 - b. Right-click **Resource Files** and select **Add Files**.
 - c. Navigate to your WebSphere Application Server 4.0 EAR file and click **Select**.
 - d. In the upper left pane of the Add Files window, navigate to your WAR file.
 - e. Select each of the directories and files in the WAR file, excluding `META-INF` and `WEB-INF`, and click **Add**.
 - f. After you add all of the required resource files, click **OK**.
6. Import your Web components.
 - a. Right-click **Web Components** and select **Import**.
 - b. In the Import Components window click **Browse**.
 - c. Navigate to your WebSphere Application Server 4.0 EAR file and click **Open**.
 - d. In the left top pane of the **Import Components** window, highlight the WAR file that you are migrating.
 - e. Highlight each of the components that display in the right top pane and click **Add**.
 - f. When all of your Web components display in the Selected Components pane of the window, click **OK**.
 - g. Verify that your Web components are correctly imported under the Web Components section of your new Web module.
7. Add servlet mappings for each of your Web components.
 - a. Right-click **Servlet Mappings** and select **New**.
 - b. Identify a URL pattern for the Web component.
 - c. Select the web component from the Servlet drop-down box.
 - d. Click **OK**.

8. Add any necessary resource references by following the instructions in the *Developing and Deploying Applications* PDF.
9. Add any other Web module properties that are required. Click **Help** for a description of the settings.
10. **Save** the Web module.

Converting a 1.1 EJB module to a 2.1 EJB module (or later)

Use the following steps to migrate each of your EJB modules.

1. Open an assembly tool.
2. Create a new EJB Module by selecting **File > New > EJB Module**.
3. Add any required class files to the new module.
 - a. Right-click **Files object** and select **Add Files**.
 - b. In the Add Files window click **Browse**.
 - c. Navigate to your WebSphere Application Server 4.0 EAR file and click **Select**.
 - d. In the upper left pane of the Add Files window, navigate to your enterprise bean JAR file.
 - e. Select each of the directories and class files and click **Add**.
 - f. After you add all of the required class files, click **OK**.
4. Create your session beans and entity beans. To find help on this subject, see Migrating enterprise bean code to the supported specification, the documentation for Rational Application Developer, or the documentation for WebSphere Studio Application Developer Integration Edition.
5. Add any necessary resource references by following the instructions in the *Developing and Deploying Applications* PDF.
6. Add any other EJB module properties that are required. Click **Help** for a description of the settings.
7. **Save** the EJB module.
8. Generate the deployed code for the EJB module by clicking **File > Generate Code for Deployment**.
9. Fill in the appropriate fields and click **Generate Now**.

Add the EJB modules and Web modules to an EAR file

1. Open an assembly tool.
2. Create a new Application by selecting **File > New > Application**.
3. Add each of your EJB modules.
 - a. Right-click **EJB Modules** and select **Import**.
 - b. Navigate to your converted EJB module and click **Open**.
 - c. Click **OK**.
4. Add each of your Web modules.
 - a. Right-click **Web Modules** and select **Import**.
 - b. Navigate to your converted Web module and click **Open**.
 - c. Fill in a **Context root** and click **OK**.
5. Identify any other application properties. Click **Help** for a description of the settings.
6. Save the EAR file.

Installing the Application on WebSphere Application Server

1. Create a JDBC provider and a data source object following the instructions in the *Developing and Deploying Applications* PDF.
2. Install the application, following the instructions in the *Developing and Deploying Applications* PDF and bind the resource references to the data source that you created.

Connection considerations when migrating servlets, JavaServer Pages, or enterprise session beans

Because WebSphere Application Server provides backward compatibility with application modules coded to the J2EE 1.2 specification, you can continue to use Version 4 style data sources when you migrate to Application Server Version 6.x. As long as you configure Version 4 data sources *only* for J2EE 1.2 modules, the behavior of your data access application components does not change.

If you are adopting a later version of the J2EE specification along with your migration to Application Server Version 6.x, however, the behavior of your data access components can change. Specifically, this risk applies to applications that include servlets, JavaServer Page (JSP) files, or enterprise session beans that run inside local transactions over shareable connections. A behavior change in the data access components can adversely affect the use of connections in such applications.

This change affects all applications that contain the following methods:

- `RequestDispatcher.include()`
- `RequestDispatcher.forward()`
- JSP includes (`<jsp:include>`)

Symptoms of the problem include:

- Session hang
- Session timeout
- Running out of connections

Note: You can also experience these symptoms with applications that contain the components and methods described previously if you are upgrading from J2EE 1.2 modules *within* Application Server Version 6.x.

Explanation of the underlying problem

For J2EE 1.2 modules using Version 4 data sources, WebSphere Application Server issues non-shareable connections to JSP files, servlets, and enterprise session beans. All of the other application components are issued shareable connections. However, for J2EE 1.3 and 1.4 modules, Application Server issues shareable connections to *all* logically named resources (resources bound to individual references) unless you specify the connections as unshareable in the individual resource-references. Using shareable connections in this context has the following effects:

- All connections that are received and used outside the scope of a user transaction are *not* returned to the free connection pool until the encapsulating method returns, even when the connection handle issues a `close()` call.
- All connections that are received and used outside the scope of a user transaction are *not* shared with other component instances (that is, other servlets, JSP files, or enterprise beans).

For example, session bean 1 gets a connection and then calls session bean 2 that also gets a connection. Even if all properties are identical, each session bean receives its own connection.

If you do not anticipate this change in the connection behavior, the way you structure your application code can lead to excessive connection use, particularly in the cases of JSP includes, session beans that run inside local transactions over shareable connections, `RequestDispatcher.include()` routines, `RequestDispatcher.forward()` routines, or calls from these methods to other components.

Examples of the connection behavior change

Servlet A gets a connection, completes the work, commits the connection, and calls `close()` on the connection. Next, servlet A calls `RequestDispatcher.include()` to include servlet B, which performs the same steps as servlet A. Because the servlet A connection does not return to the free pool until it returns from the current method, two connections are now busy. In this way, more connections might be in use than you intended in your application. If these connections are not accounted for in the **Max Connections**

setting on the connection pool, this behavior might cause a lack of connections in the pool, which results in `ConnectionWaitTimeOut` exceptions. If the **connection wait timeout** is not enabled, or if the **connection wait timeout** is set to a large number, these threads might appear to hang because they are waiting for connections that are never returned to the pool. Threads waiting for new connections do not return the ones they are currently using if new connections are not available.

Alternatives to the connection behavior change

To resolve these problems:

1. Use unshared connections.

If you use an unshared connection and are not in a user transaction, the connection is returned to the free pool when you issue a `close()` call (assuming you commit or roll back the connection).

2. Increase the maximum number of connections.

To calculate the number of required connections, multiply the number of configured threads by the deepest level of component call nesting (for those calls that use connections).

Mail migration tip

Parts of the JavaServer Page (JSP) 1.2 specification change the way the `EmailBean` class works with `Email.jsp`.




The specifications state that the JSP container creates a JSP page implementation class for each JSP page. The name of the JSP page implementation class is implementation-dependent. The JSP page implementation object belongs to an implementation-dependent named package which can vary between one JSP and another; therefore minimal assumptions should be made. The unnamed package should not be used without explicit import of the class.

Following these specifications, you should place `EmailBean.class` in a package referred to it by the fully qualified `packageName` in `Email.jsp`. Otherwise, `Email.jsp` is unable to find `EmailBean.class`.

Migrating security configurations from previous releases

This article addresses the need to migration your security configurations from a previous release of IBM WebSphere Application Server to WebSphere Application Server, Version 6. Complete the following steps to migrate your security configurations:

- Before migrating your configurations, verify that the administrative server of the previous release is running.
 - If security is enabled in the previous release, obtain the server ID and password of the previous release. This information is needed to log onto the administrative server of the previous release during migration.
 - You can optionally disable security in the previous release before migrating the installation. There is no logon required during the installation.
1. Start the First steps wizard by launching the `firststeps.bat` or `firststeps.sh` file. The first steps file is located in the following directory:

-   `.install_root/profiles/profile_name/firststeps/firststeps.sh`
-  `install_root\profiles\profile_name\firststeps\firststeps.bat`

2. On the First steps wizard panel, click **Migration wizard**.

3. Follow the instructions provided in the First steps wizard to complete the migration.

For more information on the Migration wizard, see “Using the Migration wizard” on page 143.

The security configuration of previous WebSphere Application Server releases and its applications are migrated to the new installation of WebSphere Application Server Version 6.

This task is for migrating an installation.

If custom user registry is used in the previous version, the migration process does not migrate the class files used by the custom user registry in the `<previous_install_root>\classes` directory. Therefore, after migration, copy your custom user registry implementation classes to the `install_root\classes` directory.

If you upgrade from WebSphere Application Server, Version 5.x or 4.0.x to WebSphere Application Server, Version 6, data associated with Version 5.x or 4.0.x trust associations is not automatically migrated to Version 6. To migrate trust associations, see “Migrating trust association interceptors” on page 54.

Migrating custom user registries

Before you perform this task, it is assumed that you already have a custom user registry implemented and are working with WebSphere Application Server Version 5.x or 4.x. The custom registry in WebSphere Application Server Version 4 is based on the CustomRegistry interface. For WebSphere Application Server Version 5.x and later, the interface is called the UserRegistry interface. The WebSphere Application Server Version 4-based custom registry works without any changes to the implementation in WebSphere Application Server Version 5.x or later except when the implementation is using data sources to connect to a database during initialization. If the previous implementation is using a data source to access a database, change the implementation to use JDBC connections to connect to the database. The WebSphere Application Server Version 4 version of the CustomRegistry interface was deprecated in WebSphere Application Server Version 5. So, moving your implementation to the WebSphere Application Server Version 5.x and later based interface is expected.

In WebSphere Application Server Version 5.x and later, in addition to the UserRegistry interface, the custom user registry requires the Result object to handle user and group information. This file is already provided in the package and you are expected to use it for the getUsers, getGroups and the getUsersForGroup methods.

In WebSphere Application Server Version 4.x, it might have been possible to use other WebSphere Application Server components (for example, datasources) to initialize the custom registry. This is no longer possible in WebSphere Application Server Version 5 or later, because other components like the containers are initialized after security and are not available during the registry initialization. In WebSphere Application Server Version 5, a custom registry implementation is a pure custom implementation, independent of other WebSphere Application Server components.

In WebSphere Application Server Version 4, if you had display names for users the EJB method getCallerPrincipal() and the servlet methods getUserPrincipal() and getRemoteUser() returned the display names. This behavior changed in WebSphere Application Server Version 5.x. By default, these methods now return the security name instead of the display name. However, if you need the display names to return, set the WAS_UseDisplayName property to **true**. See the getUserDisplayName method description or the Javadoc, for more information.

If the migration tool was used to migrate the WebSphere Application Server Version 4 configuration to WebSphere Application Server Version 5.x or later, be aware that this migration does not involve any changes to your existing code. Since the WebSphere Application Server Version 4 custom registry works in WebSphere Application Server Version 5.x or later without any changes to the implementation (except when using data sources) you can use the Version 4-based custom registry after the migration without modifying the code. Consider that the migration tool might not copy your implementation files from Version 4 to Version 5.x or later. You might have to copy them to the class path in the Version 6 setup (preferably to the classes subdirectory, just like in Version 4). If you are using the WebSphere Application Server Network Deployment version, copy the files to the cell and to each of the nodes class paths.

In Version 5.x or later, a case insensitive authorization can occur when using the custom registry. This authorization is in effect only on the authorization check. This function is useful in cases where your custom registry returns inconsistent (in terms of case) results for user and group unique IDs.

Note: Setting this flag does not have any effect on the user names or passwords. Only the unique IDs returned from the registry are changed to lowercase before comparing them with the information in the authorization table, which is also converted to lowercase during run time.

Before proceeding, look at the UserRegistry interface. See the *Security Guide* PDF for a description of each of these methods in detail and the changes from Version 4.

The following steps go through in detail all the changes required to move your WebSphere Application Server Version 4 custom user registry to the Version 5.x or later custom user registry. The steps are very simple and involve minimal code changes. The sample implementation file is used as an example when describing some of the steps.

1. Change your implementation to UserRegistry instead of CustomRegistry. Change:

```
public class FileRegistrySample implements CustomRegistry
to
public class FileRegistrySample implements UserRegistry
```

2. Throw the java.rmi.RemoteException in the constructors public FileRegistrySample() throws java.rmi.RemoteException
3. Change the mapCertificate method to take a certificate chain instead of a single certificate. Change

```
public String mapCertificate(X509Certificate cert)
to
public String mapCertificate(X509Certificate[] cert)
```

Having a certificate chain gives you the flexibility to act on the chain instead of one certificate. If you are only interested in the first certificate just take the first certificate in the chain before processing. In Version 5, the mapCertificate method is called to map the user in a certificate to a valid user in the registry, when certificates are used for authentication by the Web or the Java clients (transport layer certificates, Identity Assertion certificates). In Version 4, this was only called by Web clients since the Common Secure Interoperability Version 2 (CSIv2) protocol was not supported.

4. Remove the getUsers() method.
5. Change the signature of the getUsers(String) method to return a Result object and accept an additional parameter (int). Change:

```
public List getUsers(String pattern)
to
public Result getUsers(String pattern, int limit)
```

In your implementation, construct the Result object from the list of the users obtained from the registry (whose number is limited to the value of the limit parameter) and call the setHasMore() method on the Result object if the total number of users in the registry exceeds the limit value.

6. Change the signature of the getUsersForGroup(String) method to return a Result object and accept an additional parameter (int) and throw a new exception called NotImplementedException. Change the following:

```
public List getUsersForGroup(String groupName)
    throws CustomRegistryException,
           EntryNotFoundException {
```

to

```
public Result getUsersForGroup(String groupSecurityName, int limit)
    throws NotImplementedException,
        EntryNotFoundException,
        CustomRegistryException {
```

In Version 5.x and later, this method is not called directly by the WebSphere Application Server Security component. However, other components of the WebSphere Application Server like the WebSphere Business Integration Server Foundation Process Choreographer use this method when staff assignments are modeled using groups. Since this already is implemented in WebSphere Application Server Version 4, it is recommended that you change the implementation similar to the `getUsers` method as explained in step 5.

7. Remove the `getUniqueUserIds(String)` method.
8. Remove the `getGroups()` method.
9. Change the signature of the `getGroups(String)` method to return a `Result` object and accept an additional parameter (`int`). change the following:

```
public List getGroups(String pattern)
```

to

```
public Result getGroups(String pattern, int limit)
```

In your implementation, construct the `Result` object from the list of the groups obtained from the registry (whose number is limited to the value of the `limit` parameter) and call the `setHasMore()` method on the `Result` object if the total number of groups in the registry exceeds the `limit` value.

10. Add the `createCredential` method. This method is not called at this time, so return as `null`.

```
public com.ibm.websphere.security.cred.WSCredential
    createCredential(String userSecurityName)
        throws CustomRegistryException,
            NotImplementedException,
            EntryNotFoundException {
    return null;
}
```

The first and second lines of the previous code example normally appear on one line. However, it extended beyond the width of the page.

11. To build the Version 5.x and later implementation make sure you have the `sas.jar` and `wssec.jar` in your class path.

```
%install_root%\java\bin\javac -classpath %WAS_HOME%\lib\wssec.jar;
%WAS_HOME%\lib\sas.jar FileRegistrySample.java
```

Type the previous lines as one continuous line.

To build the Version 4 custom registry in Version 5.x and later, only the `sas.jar` file is required.

12. Copy the implementation classes to the product class path. The `%install_root%/lib/ext` directory is the preferred location.
13. Use the administrative console to set up the custom registry. Follow the instructions in the *Security Guide* PDF to set up the custom registry including the `IgnoreCase` flag. Make sure that you add the `WAS_UseDisplayName` properties, if required.

Migrates a Version 4 custom registry to the Version 5.x and later custom registry.

This step is required to migrate a custom registry from WebSphere Application Server Version 4 to WebSphere Application Server Version 5.x and later.

If you are enabling security, make sure you complete the remaining steps. Once completed, save the configuration and restart all the servers. Try accessing some J2EE resources to verify that the custom registry migration was successful.

Migrating trust association interceptors

The following topics are addressed in this document:

- Changes to the product-provided trust association interceptors
- Migrating product-provided trust association interceptors
- Changes to the custom trust association interceptors
- Migrating custom trust association interceptors

Changes to the product-provided trust association interceptors

For the product provided implementation for the WebSeal server a new optional property `com.ibm.websphere.security.webseal.ignoreProxy` has been added. If this property is set to `true` or `yes`, the implementation does not check for the proxy host names and the proxy ports to match any of the host names and ports listed in the `com.ibm.websphere.security.webseal.hostnames` and the `com.ibm.websphere.security.webseal.ports` property respectively. For example, if the VIA header contains the following information:

```
HTTP/1.1 Fred (Proxy), 1.1 Sam (Apache/1.1),  
HTTP/1.1 webseal1:7002, 1.1 webseal2:7001
```

Note: The previous VIA header information was split onto two lines due to the width of the printed page.

and the `com.ibm.websphere.security.webseal.ignoreProxy` is set to `true` or `yes`, the host name Fred is not be used when matching the host names. By default, this property is not set, which implies that any proxy host names and ports expected in the VIA header should be listed in the host names and the ports properties to satisfy the `isTargetInterceptor` method.

Migrating product-provided trust association interceptors

The properties located in the `webseal.properties` and `trustedserver.properties` files are not migrated from previous versions of the WebSphere Application Server. You must migrate the appropriate properties to WebSphere Application Server, Version 5 using the trust association panels in the administrative console.

Changes to the custom trust association interceptors

If the custom interceptor extends, `com.ibm.websphere.security.WebSphereBaseTrustAssociationInterceptor`, then implement the following new method to initialize the interceptor:

```
public int init (java.util.Properties props);
```

WebSphere Application Server checks the return status before using the Trust Association implementation. Zero (0) is the default value for indicating the interceptor was successfully initialized.

However, if a previous implementation of the trust association interceptor returns a different error status you can either change your implementation to match the expectations or make one of the following changes:

Method 1:

Add the `com.ibm.websphere.security.trustassociation.initState` property in the trust association interceptor custom properties. Set the property to the value that indicates that the

interceptor is successfully initialized. All of the other possible values imply failure. In case of failure, the corresponding trust association interceptor is not used.

Method 2:

Add the `com.ibm.websphere.security.trustassociation.ignoreInitStatus` property in the trust association interceptor custom properties. Set the value of this property to **true**, which tells WebSphere Application Server to ignore the status of this method. If you add this property to the custom properties, WebSphere Application Server does not check the return status, which is similar to previous versions of WebSphere Application Server.

The `public int init (java.util.Properties props);` method replaces the `public int init (String propsFile)` method.

The `init(Properties)` method accepts a `java.util.Properties` object which contains the set of properties required to initialize the interceptor. All the properties set for an interceptor (by using the Custom Properties link for that interceptor or using scripting) will be sent to this method. The interceptor can then use these properties to initialize itself. For example, in the product provided implementation for the WebSEAL server, this method reads the hosts and ports so that a request coming in can be verified to come from trusted hosts and ports. A return value of 0 implies that the interceptor initialization is successful. Any other value implies that the initialization was not successful and the interceptor will not be used.

All the properties set for an interceptor (by using the **Custom Properties** link in the administrative console for that interceptor or using scripting) is sent to this method. The interceptor can then use these properties to initialize itself. For example, in the product-provided implementation for the WebSEAL server, this method reads the hosts and ports so that an incoming request can be verified to come from trusted hosts and ports. A return value of 0 implies that the interceptor initialization is successful. Any other value implies that the initialization was not successful and the interceptor is ignored.

Note: The `init(String)` method still works if you want to use it instead of implementing the `init(Properties)` method. The only requirement is that the file name containing the custom trust association properties should now be entered using the **Custom Properties** link of the interceptor in the administrative console or by using scripts. You can enter the property using *either* of the following methods. The first method is used for backward compatibility with previous versions of WebSphere Application Server.

Method 1:

The same property names used in the previous release are used to obtain the file name. The file name is obtained by concatenating the `.config` to the `com.ibm.websphere.security.trustassociation.types` property value. If the file name is called `myTAI.properties` and is located in the `C:/WebSphere/AppServer/properties` directory, set the following properties:

- `com.ibm.websphere.security.trustassociation.types = myTAItype`
- `com.ibm.websphere.security.trustassociation.myTAItype.config = C:/WebSphere/AppServer/properties/myTAI.properties`

Method 2:

You can set the `com.ibm.websphere.security.trustassociation.initPropsFile` property in the trust association custom properties to the location of the file. For example, set the following property:

```
com.ibm.websphere.security.trustassociation.initPropsFile=  
C:/WebSphere/AppServer/properties/myTAI.properties
```

The previous line of code was split into two lines due to the width of the screen. Type as one continuous line.

However, it is highly recommended that your implementation be changed to implement the `init(Properties)` method instead of relying on `init (String propsfile)` method.

Migrating custom trust association interceptors

The trust associations from previous versions of WebSphere Application Server are not automatically migrated to version 5.x and later. Users can manually migrate these trust associations using the following steps:

1. Recompile the implementation file, if necessary.

For more information, refer to the "Changes to the custom trust association interceptors" section previously discussed in this document.

To recompile the implementation file, type the following:

```
%WAS_HOME%/java/bin/javac -classpath %WAS_HOME%/lib/wssec.jar;  
%WAS_HOME%/lib/j2ee.jar <your implementation file>.java
```

Note: The previous line of code was broken into two lines due to the width of the page. Type the code as one continuous line.

2. Copy the custom trust association interceptor class files to a location in your product class path. It is suggested that you copy these class files into the %WAS_HOME%/lib/ext directory.
3. Start the WebSphere Application Server
4. Enable security to use the trust association interceptor. *The properties located in your custom trust association properties file and in the trustedserver.properties file are not migrated from previous versions of WebSphere Application Server to version 5. You must migrate the appropriate properties to WebSphere Application Server, version 5.x or later using the trust association panels in the GUI.*

Migrating Common Object Request Broker Architecture programmatic login to Java Authentication and Authorization Service

WebSphere Application Server fully supports the Java Authentication and Authorization Service (JAAS) as programmatic login APIs. See the *Security Guide* PDF for more details on JAAS support.

This document outlines the deprecated Common Object Request Broker Architecture (CORBA) programmatic login APIs and the alternatives provided by JAAS. The following are the deprecated CORBA programmatic login APIs:

- `${user.install.root}/installedApps/sampleApp.ear/default_app.war/WEB-INF/classes/LoginHelper.java`.

The sampleApp is not included in Version 5.x and later.

- `${user.install.root}/installedApps/sampleApp.ear/default_app.war/WEB-INF/classes/ServerSideAuthenticator.java`.

The sampleApp is not included in Version 5.x and later.

- **com.ibm.IExtendedSecurity.LoginHelper.**

This API is included with the product, but is deprecated.

- **org.omg.SecurityLevel2.Credentials.**

This API is included with the product, but not recommended to use.

The APIs provided in WebSphere Application Server Version 5.x and later are a combination of standard JAAS APIs and a product implementation of standard JAAS interfaces.

The following information is only a summary; refer to the JAAS documentation for your platform located at: <http://www.ibm.com/developerworks/java/jdk/security/>.

- Programmatic login APIs:
 - `javax.security.auth.login.LoginContext`
 - `javax.security.auth.callback.CallbackHandler` interface: The WebSphere Application Server product provides the following implementation of the `javax.security.auth.callback.CallbackHandler` interface:

com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl

Provides a non-prompt CallbackHandler when the application pushes basic authentication data (user ID, password, and security realm) or token data to product LoginModules. This API is recommended for server-side login.

com.ibm.websphere.security.auth.callback.WSGUICallbackHandlerImpl

Provides a login prompt CallbackHandler to gather basic authentication data (user ID, password, and security realm). This API is recommended for client-side login.

Note: If this API is used on the server side, the server is blocked for input.

- javax.security.auth.callback.Callback interface:

javax.security.auth.callback.NameCallback

Provided by JAAS to pass the user name to the LoginModules interface.

javax.security.auth.callback.PasswordCallback

Provided by JAAS to pass the password to the LoginModules interface.

com.ibm.websphere.security.auth.callback.WSCredTokenCallbackImpl

Provided by the product to perform a token-based login. With this API, an application can pass a token-byte array to the LoginModules interface.

- javax.security.auth.spi.LoginModule interface

WebSphere Application Server provides LoginModules implementation for client and server-side login. Refer to the *Security Guide* PDF for details.

- javax.security.Subject:

com.ibm.websphere.security.auth.WSSubject

An extension provided by the product to invoke remote J2EE resources using the credentials in the javax.security.Subject

com.ibm.websphere.security.cred.WSCredential

After a successful JAAS login with the WebSphere Application Server LoginModules interfaces, a com.ibm.websphere.security.cred.WSCredential credential is created and stored in the Subject.

com.ibm.websphere.security.auth.WSPrincipal

An authenticated principal, that is created and stored in a Subject that is authenticated by the WebSphere LoginModules interface.

1. Use the following as an example of how to perform programmatic login using the CORBA-based programmatic login APIs: The CORBA-based programmatic login APIs are replaced by JAAS login.

```
public class TestClient {
    ...
    private void performLogin() {
        // Get the ID and password of the user.
        String userid = customGetUserid();
        String password = customGetPassword();

        // Create a new security context to hold authentication data.
        LoginHelper loginHelper = new LoginHelper();
        try {
            // Provide the ID and password of the user for authentication.
            org.omg.SecurityLevel2.Credentials credentials =
                loginHelper.login(userid, password);

            // Use the new credentials for all future invocations.
            loginHelper.setInvocationCredentials(credentials);
            // Retrieve the name of the user from the credentials
            // so we can tell the user that login succeeded.

            String username = loginHelper.getUserName(credentials);
            System.out.println("Security context set for user: "+username);
        } catch (org.omg.SecurityLevel2.LoginFailed e) {
            // Handle the LoginFailed exception.
        }
    }
    ...
}
```

- Use the following example to migrate the CORBA-based programmatic login APIs to the JAAS programmatic login APIs. The following example assumes that the application code is granted for the required Java 2 security permissions. For more information, see the *Security Guide* PDF and the JAAS documentation located at: <http://www.ibm.com/developerworks/java/jdk/security/>.

```

public class TestClient {
    ...
    private void performLogin() {
        // Create a new JAAS LoginContext.
        javax.security.auth.login.LoginContext lc = null;

        try {
            // Use GUI prompt to gather the BasicAuth data.
            lc = new javax.security.auth.login.LoginContext("WSLogin",
                new com.ibm.websphere.security.auth.callback.WSGUICallbackHandlerImpl());

            // create a LoginContext and specify a CallbackHandler implementation
            // CallbackHandler implementation determine how authentication data is collected
            // in this case, the authentication data is collected by login prompt
            // and pass to the authentication mechanism implemented by the LoginModule.
        } catch (javax.security.auth.login.LoginException e) {
            System.err.println("ERROR: failed to instantiate a LoginContext and the exception: "
                + e.getMessage());
            e.printStackTrace();

            // may be javax.security.auth.AuthPermission "createLoginContext" is not granted
            // to the application, or the JAAS Login Configuration is not defined.
        }

        if (lc != null)
            try {
                lc.login(); // perform login
                javax.security.auth.Subject s = lc.getSubject();
                // get the authenticated subject

                // Invoke a J2EE resources using the authenticated subject
                com.ibm.websphere.security.auth.WSSubject.doAs(s,
                    new java.security.PrivilegedAction() {
                        public Object run() {
                            try {
                                bankAccount.deposit(100.00); // where bankAccount is an protected EJB
                            } catch (Exception e) {
                                System.out.println("ERROR: error while accessing EJB resource, exception: "
                                    + e.getMessage());
                                e.printStackTrace();
                            }
                            return null;
                        }
                    }
                );

                // Retrieve the name of the principal from the Subject
                // so we can tell the user that login succeeded,
                // should only be one WSPPrincipal.
                java.util.Set ps =
                    s.getPrincipals(com.ibm.websphere.security.auth.WSPPrincipal.class);
                java.util.Iterator it = ps.iterator();
                while (it.hasNext()) {
                    com.ibm.websphere.security.auth.WSPPrincipal p =
                        (com.ibm.websphere.security.auth.WSPPrincipal) it.next();
                    System.out.println("Principal: " + p.getName());
                }
            } catch (javax.security.auth.login.LoginException e) {
                System.err.println("ERROR: login failed with exception: " + e.getMessage());
                e.printStackTrace();

                // login failed, might want to provide relogin logic
            }
        }
    }
}

```

```
}  
}  
...  
}
```

Migrating from the CustomLoginServlet class to servlet filters

The CustomLoginServlet class was deprecated in Version 5. Those applications using the CustomLoginServlet class to perform authentication now need to use form-based login. Using the form-based login mechanism, you can control the look and feel of the login screen. In form-based login, a login page is specified that displays when retrieving the user ID and password information. You also can specify an error page that displays when authentication fails.

If login and error pages are not enough to implement the CustomLoginServlet class, use servlet filters. Servlet filters can dynamically intercept requests and responses to transform or use the information contained in the requests or responses. One or more servlet filters attach to a servlet or a group of servlets. Servlet filters also can attach to JSP files and HTML pages. All the attached servlet filters are called before invoking the servlet.

Both form-based login and servlet filters are supported by any Servlet 2.3 specification-compliant Web container. A form login servlet performs the authentication and servlet filters can perform additional authentication, auditing, or logging tasks.

To perform pre-login and post-login actions using servlet filters, configure these servlet filters for either form login page or for /j_security_check URL. The j_security_check is posted by the form login page with the j_username parameter, containing the user name and the j_password parameter containing the password. A servlet filter can use user name and password information to perform more authentication or meet other special needs.

1. Develop a form login page and error page for the application, as described in the *Security Guide* PDF.
2. Configure the form login page and the error page for the application as described in the *Security Guide* PDF.
3. Develop servlet filters if additional processing is required before and after form login authentication. Refer to the *Security Guide* PDF for details.
4. Configure the servlet filters developed in the previous step for either the form login page URL or for the /j_security_check URL. Use an assembly tool or development tools like Rational Application Developer to configure filters. After configuring the servlet filters, the web-xml file contains two stanzas. The first stanza contains the servlet filter configuration, the servlet filter, and its implementation class. The second stanza contains the filter mapping section and a mapping of the servlet filter to the URL. In this case, the servlet filter maps to /j_security_check.

```
<filter id="Filter_1">  
  <filter-name>LoginFilter</filter-name>  
  <filter-class>LoginFilter</filter-class>  
  <description>Performs pre-login and post-login operation</description>  
  <init-param>  
    <param-name>ParamName</param-name>  
    <param-value>ParamValue</param-value>  
  </init-param>  
</filter>  
  
<filter-mapping>  
  <filter-name>LoginFilter</filter-name>  
  <url-pattern>/j_security_check</url-pattern>  
</filter-mapping>
```

This migration results in an application that uses form-based login and servlet filters without the use of the CustomLoginServlet class.

The use of form-based login and servlet filters by the new application are used to replace the CustomLoginServlet class. Servlet filters also are used to perform additional authentication, auditing and logging.

Migrating Java 2 security policy

Previous WebSphere Application Server releases

Starting from Version 3.x, WebSphere Application Server installed a Java 2 security manager in the server run time to prevent enterprise applications from calling the `System.exit()` and the `System.setSecurityManager()` methods. These two Java APIs have undesirable consequences if called by enterprise applications. The `System.exit()` API, for example, causes the Java virtual machine (application server process) to exit prematurely, which is an undesirable operation for an application server.

However, Java 2 security was not a fully supported feature prior to Version 5. To support Java 2 security properly, all the server run time must be marked as `privileged` (with `doPrivileged()` API calls inserted in the correct places), and identify the default permission sets or policy. Application code is not privileged and subject to the permissions defined in the policy files. The `doPrivileged` instrumentation is important and necessary to support Java 2 security. Without it, the application code must be granted the permissions required by the server run time. This is due to the design and algorithm used by Java 2 security to enforce permission checks. Please refer to the Java 2 security check permission algorithm.

The following two permissions are enforced by the WebSphere Java 2 security manager (hard coded):

- `java.lang.RuntimePermission(exitVM)`
- `java.lang.RuntimePermission(setSecurityManager)`

Application code is denied access to these permissions regardless of what is in the Java 2 security policy. However, the server run time is granted these permissions. All the other permission checks are not enforced.

Partial support was introduced since the version 4.02 product release. Prior to version 4.0.2, Java 2 security was not supported. From version 4.02 and later, only two permissions are supported:

- `java.net.SocketPermission`
- `java.net.NetPermission`

However, not all the product server run time is properly marked as `privileged`. You must grant the application code all the other permissions besides the two listed previously or the enterprise application can potentially fail to run. This Java 2 security policy for enterprise applications is liberal.

What changed

Java 2 Security is fully supported in version 5.x and later, which means all permissions are enforced. The default Java 2 security policy for enterprise application is the recommended permission set defined by the J2EE 1.4 specification. Refer to the `install_root/profiles/profile_name/config/cells/cell_name/nodes/node_name/app.policy` file for the default Java 2 security policy granted to enterprise applications. This is a much more stringent policy compared to previous releases.

All policy is declarative. The product security manager honors all policy declared in the policy files. There is an exception to this rule: enterprise applications are denied access to permissions declared in the `install_root/profiles/profile_name/config/cells/cell_name/filter.policy` file.

Note: Enterprise applications that run on Version 4.0.x with Java 2 security enabled are not guaranteed to run successfully when migrating to Version 5 (when Java 2 security is enabled), even if the Java 2 security policy is migrated properly. The default Java 2 security policy for enterprise applications is much more stringent and all permissions are enforced in Version 5. It might fail because the application code does not have the necessary permissions granted where system resources (such as file I/O for example) can be programmatically accessed and are now subject to the permission checking.

Migrating system properties

The following system properties are used in previous releases in relation to Java 2 security:

- **java.security.policy.** The absolute path of the policy file (action required). It contains both system permissions (permissions granted to the Java Virtual Machine (JVM) and the product server run time) and enterprise application permissions. Migrate the Java 2 security policy of the enterprise application to Version 5. For Java 2 security policy migration, see the steps for migrating Java 2 security policy.
- **enableJava2Security.** Used to enable Java 2 security enforcement (no action required). This is deprecated; a flag in the WebSphere configuration application programming interface (API) is used to control whether to enable Java 2 security. Enable this option through the administrative console.
- **was.home.** Expanded to the installation directory of the WebSphere Application Server (action might be required). This is deprecated; superseded by `${user.install.root}` and `${was.install.root}` properties. If the directory contains instance specific data then `${user.install.root}` is used; otherwise `${was.install.root}` is used. Use these properties interchangeably for the WebSphere Application Server or the Network Deployment environments. See the steps for migrating Java 2 security policy.

Migrating the Java 2 Security Policy

There is no easy way of migrating the Java policy file from Version 4.0.x automatically because there is a mixture of system permissions and application permissions in the same policy file. Manually copy the Java 2 security policy for enterprise applications to a `was.policy` or `app.policy` file. However, migrating the Java 2 security policy to a `was.policy` file is preferable because symbols or relative codebase is used instead of absolute codebase. There are many advantages to this process. The permissions defined in the `was.policy` file should only be granted to the specific enterprise application, while permissions in the `app.policy` file apply to all the enterprise applications running on the node where the `app.policy` file belongs. Refer to the *Security Guide* PDF for more details on policy management.

The following example illustrates the migration of a Java 2 security policy from a previous release. The contents include the Java 2 security policy file (the default is `install_root/profiles/profile_name/properties/java.policy`) for the `app1.ear` enterprise application and the system permissions (permissions granted to the JVM and product server run time). Default permissions are omitted for clarity:

```
// For product Samples
grant codeBase "file:${install_root}/installedApps/app1.ear/-" {
    permission java.security.SecurityPermission "printIdentity";
    permission java.io.FilePermission "${install_root}${/}temp${/}somefile.txt",
        "read";
};
```

For clarity of illustration, all the permissions are migrated as the application level permissions in this example. However, you can grant permissions at a more granular level at the component level (Web, enterprise beans, connector or utility Java archive (JAR) component level) or you can grant permissions to a particular component.

1. Ensure that Java 2 security is disabled on the application server.

2. Create a new `was.policy` file (if one is not present) or update the `was.policy` for migrated applications in the configuration repository in `(profiles/profile_name/config/cells/cell_name/applications/app.ear/deployments/app/META-INF/was.policy)` with the following contents:

```
grant codeBase "file:${application}" {
    permission java.security.SecurityPermission "printIdentity";
    permission java.io.FilePermission "
        ${user.install.root}${/}temp${/}somefile.txt", "read";
};
```

The third and fourth lines in the previous code sample are one continuous line, but extended beyond the width of the page.

3. Use an assembly tool to attach the `was.policy` file to the enterprise archive (EAR) file. You also can use an assembly tool to validate the contents of the `was.policy` file. For more information, see the *Security Guide* PDF.
4. Validate that the enterprise application does not require additional permissions to the migrated Java 2 Security permissions and the default permissions set declared in the `${was.install.root}profiles/profile_name/config/cells/cell_name/nodes/node_name/app.policy` file. This requires code review, code inspection, application documentation review, and sandbox testing of migrated enterprise applications with Java 2 security enabled in a pre-production environment. Refer to developer kit APIs protected by Java 2 security for information about which APIs are protected by Java 2 security. If you use third party libraries, consult the vendor documentation for APIs that are protected by Java 2 security. Verify that the application is granted all the required permissions, or it might fail to run when Java 2 security is enabled.
5. Perform pre-production testing of the migrated enterprise application with Java 2 security enabled.
Hint: Enable trace for the WebSphere Application Server Java 2 security manager in the pre-production testing environment (with trace string: `com.ibm.ws.security.core.SecurityManager=all=enabled`). This can be helpful in debugging the *AccessControlException* exception thrown when an application is not granted the required permission or some system code is not properly marked as *privileged*. The trace dumps the stack trace and permissions granted to the classes on the call stack when the exception is thrown. For more information, see the *Security Guide* PDF.

Note: Because the Java 2 security policy is much more stringent compared with previous releases, it is strongly advised that the administrator or deployer review their enterprise applications to see if extra permissions are required before enabling Java 2 security. If the enterprise applications are not granted the required permissions, they fail to run.

Migrating Version 5 Application Profiles to Version 6

The WebSphere Application Server Version 6 application profiling function works under the *unit of work* concept. This gives it a more predictable data access pattern based on the active unit of work, which could be either a transaction or an *ActivitySession*.

In order to support Java 2 platform, Enterprise Edition (J2EE) 1.3 applications with an application profile configuration from WebSphere Application Server Version 5.x, the Application Profile service on a Version 6 server must enable the Application Profiling 5.x Compatibility Mode as the default. The 5.x compatibility mode has a fair amount of performance overhead on a Version 6 server. Because of this, if there is no J2EE 1.3 application with an application profile V5.x configuration installed, the server *does not load* the support for the 5.x compatibility mode during startup, even when the 5.x compatibility mode is turned on.

After the server starts without loading the 5.x compatibility mode support, if a J2EE 1.3 application with an application profile V5.x configuration installs on the server and attempts to start, the following message is displayed, and the server must be restarted:

ACIN0031E: The J2EE 1.3 application <ApplicationName> is configured for application profiling and is installed and starting on a running server that enables Application Profiling 5.x Compatibility Mode. You must re-start the server.

This situation only happens when:

1. the server started with the Application Profile service enabled and 5.x compatibility mode turned on
2. you try to install and start a J2EE 1.3 application with an application profile configured in Version 5.x.

To avoid this situation, you must install at least one J2EE 1.3 application with an application profile Version 5.x configuration *before* starting the server.

Ideally, you would upgrade your J2EE 1.3 applications to use the Version 6 application profiling configuration and turn off the Application Profiling 5.x Compatibility Mode through the administrative console.

Therefore it is recommended that you migrate any application you might have configured with application profiling in Version 5. Application profiles migration only requires you to re-configure your applications in the Version 6 Application Server Toolkit (AST). See the *Developing and Deploying Applications* PDF for more information.

Migrating internationalized applications

Applications that used the internationalization service in WebSphere Application Server versions 4 and 5 can use the service in later versions with no modification. The packaging and structure of the internationalization context API remain identical across releases. Most importantly, the semantics of the API remain as well.

In Version 4, the internationalization service did not provide internationalization deployment descriptor policy information to direct how the service manages internationalization context across the various application components. Rather, the service employed the implicit client-side internationalization (CSI) and server-side internationalization (SSI) policies, which dictated how the service managed context according to the type of Java 2 Platform, Enterprise Edition (J2EE) container hosting a component. For details, refer to the combined information center for WebSphere Application Server Version 4. Briefly, all server components in Version 4 are SSI, and all EJB client applications are CSI.

In versions 5 and later, the internationalization type setting of all server components is configured to Container by default. The internationalization service assigns the RunAsCallercontainer internationalization attribute by default to any container-managed (CMI) servlet or enterprise bean invocation lacking a container internationalization attribute. As a result, the invocations of server components that lack internationalization policy information in the deployment descriptor run under the policy, [CMI, RunAsCaller], which is semantically equivalent to the SSI internationalization policy of Version 4; EJB client applications run under the logical policy [AMI, RunAsServer], which is equivalent to the CSI policy of Version 4.

When migrating a Version 4 application to versions 5 or later, it is unnecessary to configure the internationalization deployment descriptor information during application assembly because all component invocations run under semantically equivalent internationalization context management policies.

Chapter 5. Migrating product configurations

You can migrate administrative configurations with the Migration wizard or with the command line migration tools, as this task describes.

If you use an earlier version of WebSphere Application Server, the system administrator might have fine-tuned various application and server settings for your environment. It is important to have a strategy for migrating these settings with maximum efficiency and minimal loss.

You can perform incremental migration of V4.0.x nodes by calling the migration tools multiple times, each time specifying a different configuration file. Various reasons exist for having multiple configuration files. Whatever the reason, migrating one configuration file at a time lets you test applications incrementally before continuing to the next configuration file. You can also perform incremental migration of V5.x instances in the same manner.

Before using the migration tools, consult the V6.0 Release Notes document to understand what fixes you must apply to earlier versions. Applying fixes to an earlier version might also apply fixes to files that have a role in the migration. Apply any fixes to ensure the most effective migration of configurations and applications possible.

The migration tools in V6.0 support migration from the following versions of WebSphere Application Server: V4.0.x, V5.0.x, V5.1.x.

IBM provides a set of migration tools for migrating administrative configurations from V4.0.x, V5.0.x, or V5.1.x to the Base product.

When you use the migration tools, the overall migration process is:

1. Install the V6 product. Installing the product creates a stand-alone Application Server.
2. Start the First steps console.
3. Select the Migration wizard on the First steps console.
4. Use the Migration wizard to migrate the previous release to the V6 product.

The Migration wizard calls the `WASPostUpgrade` command line tool. `WASPostUpgrade` uses the **backupConfig** command to save the existing V6.0 configuration before performing migration. The results are stored in the `profile_name/temp` directory. You can use the **restoreConfig** command to restore the backup, if required.

1. Migrate V5.x and V4.0.x base WebSphere Application Server nodes.

Select one of the following migration scenarios for information about how to migrate configuration data from a V4.0.x or V5.x Express node or a V4.0.x or V5.x base WebSphere Application Server node to a V6 stand-alone Application Server:

- “Migrating V4.0.x of WebSphere Application Server to a V6 stand-alone Application Server” on page 66
- “Migrating V4.0.x of WebSphere Application Server to a remote stand-alone V6 machine” on page 68
- “Migrating V5.x Express to V6 Express” on page 70
- “Migrating V5.x of WebSphere Application Server to a V6 Application Server” on page 73
- “Migrating V5.x of WebSphere Application Server to a V6 stand-alone Application Server on a remote machine” on page 71
- “Migrating a V5.x Application Server configuration instance to a V6 Application Server profile” on page 73
- “Migrating from an operating system that is no longer supported” on page 74

Note: If you are migrating a V5.x managed node to a V6 managed profile, the node names must match.

2. Configure the WebSphere Application Server environment after migration, as described in “Configuring WebSphere Application Server after migration” on page 75. This is a way of verifying the results of the migration tools. You can also use “Configuration mapping during migration” on page 25 to verify the results of the migration. The topic has a detailed description of how the migration tools migrate objects, and what you should verify.

You can use the migration tools to migrate from one version of WebSphere Application Server to another.

Stand-alone migrations

Select the appropriate migration scenarios for information about how to migrate from a V4.0.x or V5.x Express node or a V4.0.x or V5.x base WebSphere Application Server node to a V6 stand-alone Application Server:

- “Migrating V4.0.x of WebSphere Application Server to a V6 stand-alone Application Server”
- “Migrating V4.0.x of WebSphere Application Server to a remote stand-alone V6 machine” on page 68
- “Migrating V5.x Express to V6 Express” on page 70
- “Migrating V5.x of WebSphere Application Server to a V6 Application Server” on page 73
- “Migrating V5.x of WebSphere Application Server to a V6 stand-alone Application Server on a remote machine” on page 71
- “Migrating a V5.x Application Server configuration instance to a V6 Application Server profile” on page 73
- “Migrating from an operating system that is no longer supported” on page 74

You can use the migration tools to migrate from one version of WebSphere Application Server to another.

Migrating V4.0.x of WebSphere Application Server to a V6 stand-alone Application Server

You can use the migration tools to migrate configuration data from Version 4.0.x of WebSphere Application Server to WebSphere Application Server V6.

If you use an earlier version of WebSphere Application Server, the system administrator might have fine-tuned various application and server settings for your environment. It is important to have a strategy for migrating these settings with maximum efficiency and minimal loss.

You can migrate administrative configurations with the Migration wizard, which is the graphical interface to the Version 6 migration tools (the “WASPreUpgrade command” on page 137 and the “WASPostUpgrade command” on page 140). You can also migrate manually, using the tools from the command line, as this task describes.

You can perform incremental manual migration by calling the migration tools multiple times, each time specifying a different configuration file. There are various reasons for having multiple configuration files. Whatever the reason, migrating one configuration file at a time lets you test applications incrementally before continuing to the next configuration file.

Manual migration provides a more incremental migration approach than the complete migration that the Migration wizard provides. IBM provides a set of migration tools for migrating administrative configurations to the base WebSphere Application Server product from Version 4.0.x. The overall migration process is to install the Version 6 product, back up the current configuration and necessary files, and restore the configuration.

Before using the migration tools, consult the Release Notes document to understand what fixes you must apply to earlier versions. Applying fixes to an earlier version might also apply fixes to files that have a role in the migration. Apply any fixes to ensure the most effective migration of configurations and applications possible.

Typically you can use the WASPreUpgrade and WASPostUpgrade migration tools to upgrade from V4.0.x to V6 on the same machine. If your scenario includes migrating a V4.0.x configuration on one machine to WebSphere Application Server V6 on another machine, use the alternate procedure described in “Migrating V4.0.x of WebSphere Application Server to a remote stand-alone V6 machine” on page 68.

This topic describes using the V6 migration tools to migrate the following products:

- WebSphere Application Server Advanced Edition, V4.0.x
- WebSphere Application Server Advanced Single Server Edition V4.0.x (the steps vary slightly)

The WASPreUpgrade tool saves the existing V4.0.x configuration into a *migration-specific-backup* directory. The WASPostUpgrade tool uses this directory to add the old configuration settings to the new V6.0 environment.

1. Perform a typical or custom V6 installation..
2. Use the V6 Profile creation wizard to create a stand-alone profile.

Note: When you create the V6 profile, the NodeName you choose must match one of the NodeNames in the V4.0.x environment.

3. **Recommended:** Use the Migration wizard to migrate the node (the recommended way to access the Migration wizard is through the First Steps console). If you choose to use the Migration wizard, you do not need to proceed through the rest of the steps in this task.
4. Save the current configuration using the WASPreUpgrade script from the *migration/bin* directory of the product CD-ROM. Save the configuration in the *migration-specific-backup* directory:

```
WASPreUpgrade /usr/tmp/migration-specific-backup /usr/websphere/appserver yourNodeName
```

For V4.0.x Advanced Edition, verify that the administrative server of the existing environment is running. The WASPreUpgrade tool provides status to the screen and to log files in the *migration-specific-backup* directory. ASCII log file names start with the text WASPreUpgrade and include a date and timestamp.

See “WASPreUpgrade command” on page 137 for a list of the files and directories from the existing V4.0.x configuration that WASPreUpgrade saves to the *backup* directory.

The WASPreUpgrade tool saves selected files from the V4.0.x *install_root/bin* directory. It also exports the existing Application Server configuration from the V4.0.x repository. The WASPreUpgrade tool calls the XMLConfig tool to export the existing V4.0.x repository to the *websphere_backup.xml* file in the *migration-specific-backup* directory.

V4.0.x Advanced Single Server Edition does not require the administrative server to run at the time of migration. The WASPreUpgrade tool copies the *server-cfg.xml* file from the *install_root/config* directory to the *migration-specific-backup/config* directory.

If errors occur for a V4.0.x Advanced Single Server Edition scenario while running the WASPreUpgrade tool, you might have to apply fixes to the V4.0.x installation to successfully complete the export step. See the IBM Support page for the latest fixes that might be applicable. When viewing this information from the information center, click **Support** to link to the IBM Support page.

5. Migrate the previous configuration to the new installation with the WASPostUpgrade tool in the *profile_name/bin* directory of the V6 installation. The WASPostUpgrade tool migrates V4.0.x configuration information created by the WASPreUpgrade tool to the V6 installation.

The WASPostUpgrade tool does not migrate Samples or the administrative console application because there are already Samples and an administrative console application in V6. The WASPostUpgrade tool records detailed information specific to each enterprise bean it deploys, in the *WASPostUpgrade.log* file.

6. Stop the administrative server of the earlier version if it is running, before running the Version 6 node.
7. Configure WebSphere Application Server after migration. This is a way of verifying the results of the migration tools. You can also use Configuration mapping during migration to verify the results of the migration. The topic has a detailed description of how the migration tools migrate objects, and what you should verify.

You can use the migration tools to perform a manual migration from Version 4.0.x of WebSphere Application Server to Version 6.

After you test and verify that the applications and configuration data you moved to the V6 node is successful, you can uninstall the V4.0.x Application Server as described in the information centers for those releases. Click the **Library** link at the bottom of any V6 information center topic to locate the information centers for the other releases.

Migrating V4.0.x of WebSphere Application Server to a remote stand-alone V6 machine

You can use the migration tools to perform a manual migration between two machines.

Typically you can use the WASPreUpgrade and the WASPostUpgrade migration tools from V6 of WebSphere Application Server to upgrade from V4.0.x to V6 on the same machine.

However, some scenarios require that you migrate the V4.0 configuration on one machine to V6 on a different machine. One of these scenarios is when you install new machines for your latest V6 environment but need to migrate your existing V4.0.x configuration from other machines.

This topic describes using the V6 migration tools to migrate the following products:

- WebSphere Application Server Advanced Edition, V4.0
- WebSphere Application Server Advanced Single Server Edition, V4.0 (the steps vary slightly)

The WASPreUpgrade tool saves the existing V4.0 configuration into a *migration-specific-backup* directory. The WASPostUpgrade tool uses this directory to add the old configuration settings to the new V6 environment.

1. Obtain the V6 product CD-ROM. On this CD is the *migration/bin* directory. This directory contains a special environment that you can use to run the WASPreUpgrade tool without installing V6.
2. Save the current configuration using the WASPreUpgrade script from the */migration/bin* directory of the V6 product CD-ROM, which you must mount to the V4.0 machine. Save the configuration in the *migration-specific-backup* directory on the V4.0 machine.

```
WASPreUpgrade /opt/tmp/migration-specific-backup /opt/websphere/appserver adminNodeName
```

For all scenarios except V4.0.x Advanced Single Server Edition, verify that the administrative server of the existing environment is running. The WASPreUpgrade tool provides status to the screen and to log files in the *migration-specific-backup* directory. ASCII log file names start with the text WASPreUpgrade and include a date and timestamp.

The WASPreUpgrade tool saves selected files from the V4.0.x */bin* directory. It also exports the existing Application Server configuration from the V4.0.x repository. The WASPreUpgrade tool calls XMLConfig to export the existing V4.0.x repository to the *websphere_backup.xml* file in the *migration-specific-backup* directory.

V4.0.x Advanced Single Server Edition does not require the administrative server to run at the time of migration. The WASPreUpgrade tool copies the *server-cfg.xml* file from the *install_root/config* directory to the *migration-specific-backup/config* directory.

If errors occur while running the WASPreUpgrade tool, you might have to apply fixes to the V4.0 installation to successfully complete the export step. See the IBM Support page for the latest fixes that might be applicable. When viewing this information from the information center, click **Support** to link to the IBM Support page.

3. Copy the *migration-specific-backup* directory from the V4.0 machine to the V6 machine. Use the **ftp** command, shared storage, or some other mechanism to copy the file to the new machine.
Perform the following steps on the machine with V6 of WebSphere Application Server.
4. Copy the *migration-specific-backup/websphere_backup.xml* or the *migration-specific-backup/config/server-cfg.xml* file and store it as an archive. You edit the original file in the next step.
5. Edit the *migration-specific-backup/websphere_backup.xml* or the */migration-specific-backup/config/server-cfg.xml* file to correct machine-dependent settings. Make the following changes in the file:
 - a. Change the node name in the *migration-specific-backup/websphere_backup.xml* file. There is no node name in the *migration-specific-backup/config/server-cfg.xml* file. If you are using the same node name for the V6 machine that you use for the original V4.0.x configuration, do not change the name. Otherwise, you must change all occurrences of the V4.0.x node name to the node name you are using for WebSphere Application Server V6. The node name occurs in many XML stanzas throughout the file. Failing to change all occurrences results in an incomplete migration of data.
 - b. Change the path names in the *migration-specific-backup/websphere_backup.xml* or the *migration-specific-backup/config/server-cfg.xml* file. The configuration file refers to path names in many XML stanzas throughout the file. Update any reference to a file outside of the V4.0.x directory structure to the equivalent directory on the new machine, even if you must create an equivalent directory. The implication of configuring a matching environment means that you might have to copy the original directory to the V6 machine. Or you might have to install the appropriate software.
 - c. Check files in the *properties* directories for references that contain path names. In particular, edit the *migration-specific-backup/properties/sas.client.props* and the *migration-specific-backup/properties/TraceSettings.properties* files to correct machine-dependent settings: Make the following changes in the file:
 - 1) Change the path values of any property in the file.
Each property file contains properties that refer to paths. Update any reference to a file outside of the V4.0.x directory structure to the equivalent directory on the new machine, even if you must create an equivalent directory.
 - 2) Correct specification styles for path values that are dependent on the operating system.
You must correct path specifications if they differ from what works on the machine running V6.
 - d. Correct specification styles for path names that are dependent on the operating system. You must correct path specifications if they differ from what works on the machine running V6. For example, if you are moving from V4.0.x on a Windows platform to V6 on a Linux platform, change any Windows-specific path in the configuration file to use the Linux path style. Change `c:\mystuff\somepath` to `/opt/mystuff/somepath`.
 - e. Change user IDs and passwords to match security requirements. You might have to change user IDs and passwords if they are not identical to those in use on the V6 machine.
To change an encoded password to a clear-text password, change `<password>{xor}LCoxayht</password>` to `<password>mypassword</password>`.
 - f. Change other machine-specific information. The configuration might refer to other software products or configurations that do not exist on the new machine. For example, the old machine might have a database. The V6 configuration should still point to the database on the old machine, possibly. Modify the data source to point to the database on the V4.0.x machine.
6. Install the V6 product.
7. Use the Profile Creation wizard to create a V6 profile.

8. Add the V4.0.x configuration to the V6 configuration. Use the WASPostUpgrade tool in the `install_root/bin` directory of V6 to add the V4.0.x configuration to the V6 configuration.

```
WASPostUpgrade /opt/tmp/migration-specific-backup
```

The WASPostUpgrade tool records detailed information specific to each enterprise bean it deploys, in the `migration-specific-backup/WASPostUpgrade.log` file.

9. Configure WebSphere Application Server after migration. This is a way of verifying the results of the migration tools. You can read Configuration mapping during migration to learn more about the results of migration. This topic has a detailed description of how the migration tools migrate objects, and what you should verify.

This procedure results in migrating WebSphere Application Server from V4.0.x to a remote V6 machine.

Migrating V5.x Express to V6 Express

You can migrate the configuration from V5.x WebSphere Application Server - Express to WebSphere Application Server - Express, V6.

This topic describes how to use the V6 WebSphere Application Server migration tools to migrate your applications and configuration from V5 WebSphere Application Server - Express to V6 WebSphere Application Server - Express.

1. Use the V6 Migration wizard to transfer configuration settings from V5 Express to V6 Express.
 - a. Start the First steps console at the end of installation, or by issuing the following command from the `firststeps` directory:

-  `.install_root/firststeps/firststeps.sh`

-  `install_root\firststeps\irststeps.bat`

- b. Start the Migration wizard from the First steps console.

The Migration wizard uses the WASPreUpgrade command line tool and the WASPostUpgrade command line tool to migrate the data and applications from V5 to V6.

This step transfers configuration information for the V5 Express server resources, security, variables, and virtual hosts to the V6 Express product. All stored information is in XML files in the `install_root/profiles/profile_name/config/cells` directory of each product. For a full description of each XML file, see the Configuration documents and Configuration document descriptions topics in the information center.

The WASPreUpgrade command line tool saves selected files from the `/bin` directory to a backup directory you specify on a wizard panel. Migration saves files from the following directories to the backup directory:

- `classes` (This directory is not saved for iSeries platforms.)
- `config`
- `installableApps`
- `installedApps` (or an alternate directory specified by the user)
- `properties` (A subset of files in this directory is saved for iSeries platforms.)

Later, the Migration wizard uses the WASPostUpgrade migration tool to restore the environment in the backup directory into the V6 Express installation.

2. Resolve port differences.

The Migration wizard displays a list of ports for Express. If the same port is used for more than one purpose, the migration tools log a message that states you must manually resolve the port conflict.

Two groups of port differences can remain within port assignments in the virtual hosts file, at the cell level:

- **HTTP transport chains**

WebSphere Application Server and Express use different HTTP transport ports by default. To maintain port assignments used by Express when migrating to the base WebSphere Application Server product, add the Express port values at the server level:

- a. In the WebSphere Application Server administrative console, click the **plus sign** ("+") next to **Servers**
- b. Click **Application Servers**.
- c. Click the server on which you plan to install your applications.
- d. Click **Web Container settings**.
- e. Click **Web container transport chains**.
- f. Use Studio Site Developer to find the original V5 Express remote server HTTP transport ports:
 - 1) Display the **Server Configuration** view.
 - 2) Double-click the Express remote server that you are migrating.
 - 3) Click **Ports**.
 - 4) View ports defined under **Server Settings**.
- g. Click **New** in the V6 WebSphere Application Server administrative console to create additional entries needed to define a port.

- **Advanced ports**

There are a number of special purpose ports defined for Express. These ports are described in the "Migrating and coexisting" on page 23 topic. Use non-conflicting values to continue to run both V5 Express and V6 Express on the same system (coexistence), or if WebSphere Application Server applications are dependent on the port values.

Using V6 Express port values might require you to change applications. Change any references to port values that are changing. To identify Express applications that require port changes, modify the port configuration on the V5 Express installation to use V6 port values. Test the applications that you intend to migrate to identify port references that are in error.

- a. In the WebSphere Application Server administrative console, click the **plus sign** ("+") next to **Servers**.
- b. Click **Application Servers** to get a list of servers.
- c. Click the server on which you intend to install the applications.
- d. Click **Ports**.
- e. Use Studio Site Developer to find the original V5 Express remote server port assignments:
 - 1) Display the **Server Configuration** view.
 - 2) Double click the Express remote server that you are migrating.
 - 3) Click **Ports**.
 - 4) View **Advanced Ports** defined under **Server Settings**.

3. Verify migrated applications.

To verify that applications are installed successfully:

- a. Click **Enterprise Applications** in the administrative console.
- b. Check your migrated applications.
- c. Click **Start**.

Verify that your applications start successfully.

You can successfully migrate configuration files and applications from V5 Express to V6 Express.

Migrating V5.x of WebSphere Application Server to a V6 stand-alone Application Server on a remote machine

You can use the migration tools to migrate between two machines.

Typically you can use the WASPreUpgrade and the WASPostUpgrade migration tools to upgrade from Version 5.x to Version 6 on the same machine.

However, some scenarios require that you migrate the V5.x configuration on one machine to V6 on a different machine. One of these scenarios is when you install new machines for your V6 environment but need to migrate your existing V5.x configuration from other machines.

To ensure that your operating system is supported by WebSphere Application Server Version 6, visit the following site for the most current list of supported hardware and software:

- WebSphere Application Server system requirements

If you find that your operating system is not supported for migration to V6, see “Migrating from an operating system that is no longer supported” on page 74.

This task describes how to use the V6 migration tools to migrate WebSphere Application Server, Version 5.x to Version 6 on a separate machine.

The WASPreUpgrade tool saves the existing V5.x configuration into a *migration-specific-backup* directory. The WASPostUpgrade tool uses this directory to add the old configuration settings to the new V6 environment.

1. Obtain the V6 product CD-ROM. On this CD is the *migration/bin* directory. This directory contains a special environment that you can use to run the WASPreUpgrade tool without installing V6.
2. Save the current configuration using the WASPreUpgrade script from the *migration/bin* directory of the V6 product CD-ROM, which you must mount to the V5.x machine. Save the configuration in the *migration-specific-backup* directory on the V5.x machine.

Linux:

```
./WASPreUpgrade.sh /filepath/migration-specific-backup /opt/WebSphere/AppServer
```

Windows:

```
WASPreUpgrade C:\filepath\migration-specific-backup C:\Program Files\WebSphere\AppServer
```

The WASPreUpgrade tool provides status to the screen and to log files in the *migration-specific-backup* directory. ASCII log file names start with the text WASPreUpgrade and include a date and timestamp.

3. Copy the *migration-specific-backup* directory from the V5.x machine to the V6 machine. Use the **ftp** command, shared storage, or some other mechanism to copy the directory to the new machine.
4. **Optional:** Run the backupConfig tool from the 5.x *install_root* to create a backup of your 5.x environment.
5. Install V6 of WebSphere Application Server. Install the same features as the earlier release.
6. Create a profile with the Profile Creation wizard.

Note: When you create the V6 profile, the node name you choose must match one of the NodeNames in the V4.0.x environment.

7. Add the V5.x configuration to the V6 configuration. Use the WASPostUpgrade tool in the *install_root/bin* directory of the V6 installation to add the V5.x configuration to the V6 configuration.

Linux:

```
./WASPostUpgrade.sh /filepath/migration-specific-backup
```

Windows:

```
WASPostUpgrade C:\filepath\migration-specific-backup
```

The WASPostUpgrade tool records detailed information specific to each enterprise bean it deploys, in the *migration-specific-backup/WASPostUpgrade.log* file.

8. Modify the configuration using the WebSphere Application Server 6 administration interfaces. Make these changes:
 - a. Change user IDs and passwords to match security requirements. You might have to change user IDs and passwords if they are not identical to those in use on the V5.x machine.
 - b. Change other machine-specific information. The configuration might refer to other software products or configurations that do not exist on the new machine. For example, the old machine might have a database. Modify the data source to point to the database on the old machine.

You have migrated WebSphere Application Server from V5.x to a remote V6 machine.

Migrating V5.x of WebSphere Application Server to a V6 Application Server

The migration tools of WebSphere Application Server, Version 6 support migrating configuration data from V5.x to V6.

If your scenario includes migrating a V5.x configuration on one machine to WebSphere Application Server V6 on another machine, use the alternate procedure described in “Migrating V5.x of WebSphere Application Server to a V6 stand-alone Application Server on a remote machine” on page 71.

This task describes how to migrate the configuration of a V5.x node to V6. This task describes how to use the V6 migration tools to perform the migration.

1. Stop all of the V5.x Application Servers that are running on the node. Use the **stopServer** command from the *install_root/bin* directory. For example, issue the following commands to stop server1 and server2 on a Linux platform:

```
stopServer.sh server1
stopServer.sh server2
```

If security is enabled, specify the **-user** and **-password** parameters on the **stopServer** command.

You can migrate a V5.x node without stopping it. It is not necessary to have the node running to migrate its configuration. The migration tools can retrieve all the configuration data while the node is stopped. You must stop the node before you can start the V6 node that you are installing.

2. Install the V6 product.
3. Use the Migration wizard to migrate the node (the recommended way to access the Migration wizard is through the First Steps console).

After going through the Migration wizard, you have upgraded a Version 5.x node to Version 6.

Migrating a V5.x Application Server configuration instance to a V6 Application Server profile

The migration tools of WebSphere Application Server, Version 6 support migrating a V5.x configuration instance to a V6 profile.

This task describes how to migrate the configuration of a V5.x configuration instance to a V6 profile. This task describes how to use the V6 migration tools to perform the migration.

If the node you are migrating is part of a deployment manager cell, migrate the V5.x deployment manager to V6 as described in the information center for Network Deployment first, before continuing this procedure. The deployment manager node must always be at the highest release level within the cell.

1. Stop all of the V5.x Application Servers that are running on the node. Use the **stopServer** command from the *install_root/bin* directory. For example, issue the following commands to stop server1 and server2 on a Linux platform:

```
stopServer.sh server1
stopServer.sh server2
```

If security is enabled, specify the **-user** and **-password** parameters on the **stopServer** command.

You can migrate a V5.x node without stopping it. But it is not necessary to have the node running to migrate its configuration. The migration tools can retrieve all the configuration data while the node is stopped. You must stop the node before you can start the V6 node that you are installing.

2. Install the V6 product.
3. From the *profile_name/bin* directory of the V6 node, run the following command:
UNIX:

```
./WASPreUpgrade.sh <backupDirectory> <V5_location>
```

Windows:

```
WASPreUpgrade.bat <backupDirectory> <V5_location>
```

where *backupDirectory* is the name of the directory you want the output of WASPreUpgrade to be stored, and *V5_location* is the location where the V5.x node is installed.

4. From the V6 node's <profile_name>/bin directory, run the following command:

UNIX:

```
./WASPostUpgrade.sh <backupDirectory> -oldProfile <host_instanceName> -profileName <ProfileName>
```

Windows:

```
WASPostUpgrade.bat <backupDirectory> -oldProfile <host_instanceName> -profileName <ProfileName>
```

where *backupDirectory* is the same directory you specified in WASPreUpgrade, and *ProfileName* is the name of the V6 profile. (See "WASPostUpgrade command" on page 140 for more information on WASPostUpgrade.)

5. Start the V6 node.
6. **Optional:** Uninstall the V5.x node. Perform this step only after you are certain that you have successfully migrated the configuration of the node you intend to delete.

You have upgraded a Version 5.x configuration instance to a Version 6 profile.

Migrating from an operating system that is no longer supported

You can migrate an earlier WebSphere Application Server Version 4.0.x or Version 5.x release that is running on an operating system that Version 6.0 does not support.

1. Start up the WebSphere Application Server Version 4.0.x or Version 5.x Administrative Server.
2. Run the WASPreUpgrade command line migration tool. Run the command from the migration\bin (or migration/bin) directory in the *platform_root* of the Version 6 CD-ROM.
3. Shut down the WebSphere Application Server Version 4.0.x or Version 5.x release by stopping all server nodes in the configuration, including the administrative server node.
4. Tar or zip the backup directory and FTP it to another system.
5. Install the new operating system, keeping the same host name. If possible, keep the system name and passwords the same as the old system. Place any database files related to applications you are migrating in the same path as the previous system. In general, try to keep paths the same. If you do change paths and names, refer to "Migrating V5.x of WebSphere Application Server to a V6 stand-alone Application Server on a remote machine" on page 71. Make any changes before running the WASPostUpgrade command below.
6. FTP the backup directory from the other system and unzip it.
7. Install the V6 product.
See the *Getting Started* PDF for more information.
8. Run the **WASPostUpgrade** command line migration tool, from the bin directory of the Version 6 install_root. Specify the backup directory that you unzipped in step 6. See the WASPostUpgrade topic for proper command syntax.

```
install_root\bin\WASPostUpgrade WAS_HOME\migration
```

Creating a backup directory on unsupported V4.0.x operating systems

This topic applies to installations of WebSphere Application Server Advanced Server Single Edition (AEs) and Advanced Edition (AE) Version 4.0.x that are running on the AIX 4.3.3 or Solaris 7 operating systems. Due to limitations of the JDK used by WebSphere Application Server V6, you cannot use the

WASPreUpgrade command that comes with the WebSphere Application Server V6 product CD with these operating systems. To create the backup directory used as input to the WASPostUpgrade command, use the following steps.

1. **Optional:** Upgrade your operating system. If you choose to upgrade your system, you do not need to proceed to the following steps. AIX provides an "OS migration" path from release 4.3.3 to release 5.1 that retains your current system configuration. Solaris provides an "OS Upgrade" path from Solaris 7 to Solaris 8 that does the same. Consult your operating system documentation for details.
2. Manually create the backup directory:
 - a. For both AE and AEs, the first step is to back up the common set of configuration files. The following example reflects an AIX installation. You may have to update many of the values in this example to reflect your configuration requirements, or to reflect a different operating system.

```
copy /websphere/appserver/bin to /websphere/backup/websphere_backup/bin
copy /websphere/appserver/classes to /websphere/backup/websphere_backup/classes
copy /websphere/appserver/config to /websphere/backup/websphere_backup/config
copy /websphere/appserver/installableApps to /websphere/backup/websphere_backup/installableApps
copy /websphere/appserver/installedApps to /websphere/backup/websphere_backup/installedApps
copy /websphere/appserver/lib/app to /websphere/backup/websphere_backup/lib/app
copy /websphere/appserver/properties to /websphere/backup/websphere_backup/properties
```

Note: Some of the above directories may not exist on all systems.

- b. For AE only, the following additional steps are required:
 - 1) Save the following configuration files:

```
copy /websphere/appserver/installedConnectors to
   /websphere/backup/websphere_backup/installedConnectors
```
 - 2) Use the Version 4.0.x XMLConfig tool to export the current configuration. Before exporting an AE configuration to a file, be sure that the administrative server is running. See the V4.0 AE XMLConfig documentation for further details on the following parameters:
 - adminNodeName
 - export
 - nameServiceHost
 - nameServicePort

Sample export command for Version 4.0.x:

```
XMLConfig -export /websphere/backup/websphere_backup.xml -adminNodeName call
```

Configuring WebSphere Application Server after migration

Installation automatically configures IBM WebSphere Application Server, Version 6 and all other bundled products. There is no need for additional configuration if you did not migrate from an earlier version.

If you migrate an installation of WebSphere Application Server, Version 4.0.x, there are some items to review before considering your environment fully configured.

1. Examine any Lightweight Third Party Authentication (LTPA) security settings you might have used, and apply the settings in the WebSphere Application Server security settings.

Global security that uses LTPA authentication in Version 4.0.x is migrated to the base WebSphere Application Server product. However, although global security was enabled in Version 4.0.x, it is disabled during migration to Version 6.
2. Check the WASPostUpgrade.log file in the logs directory, for details about JSP 0.91 objects that the migration tools do not migrate.

Version 6 does not support JSP 0.91 objects. The migration tools do not migrate JSP objects configured to run as JSP 0.91 objects. The migration tools do, however, recognize the objects in the output and log them. Version 6 runs JSP 1.0 and 1.1 objects as JSP 1.2 objects, which is its only supported level.

3. Identify and use the migration tools to migrate non-migrated nodes in Version 4.0.x repositories that have multiple nodes.

A Version 4.0.x repository can contain more than one node name and its associated children. The WASPostUpgrade tool processes only those objects and children that match the migrating node. This determination is made by checking the names of nodes in configuration files with fully qualified and non-qualified network names of the migrating machine.

4. Update J2EE resources in client JAR files to the new resource format with the ClientUpgrade tool. J2EE applications might exist on the client, if the client has client JAR files with J2EE resources.
5. To migrate a Version 4.0.x or V5.x XML application to the Version 6 level, you must recompile it.
6. **Optional:** Configure WebSphere Application Server to use a database.
7. Review your Java virtual machine settings to verify that you are using a heap size of at least 50 for improved startup performance.

If you have used a smaller heap size in the past, you can use the default heap size, which is now 50.

Now you are finished with pre-test configuration. You might have to fine tune your WebSphere Application Server environment as you test it. Test all redeployed applications before moving them into production.

XML parser for Java code

On November 9, 1999, the Apache Software Foundation announced the creation of the xml.apache.org project for Open Source Extensible Markup Language (XML) solutions.

As part of that announcement, IBM announced that it was donating its XML4J, XML4C, and LotusXSL technologies to the xml.apache.org project. The parsing technologies have been renamed Xerces, and the LotusXSL technology has been renamed Xalan.

IBM is shifting its XML parsing development resources to work on the Xerces parsers. The objective is to use the Xerces code base as the foundation for XML4J and XML4C. This version of XML4J is based on the Apache Xerces2 code base.

What is the difference between XML4J and Xerces?

IBM has tested this version of XML4J in addition to the testing done by the xml.apache.org project. The names of the main Java archive (JAR) files have changed from xml4j.jar to xmlParserAPIs.jar and xercesImpl.jar, and from xml4jSamples.jar to xercesSamples.jar.

This version of XML4J contains one major API change and a few relatively minor API changes from the last major release. Its major features are many bugfixes and performance improvements, as well as the addition of a few features.

XML4J 4.2.x ships with the official W3C DOM HTML Recommendation API . The major change to the API was the removal of setSelected from the HTMLOptionElement interface. To workaround this change from XML4J 4.1.x, use setAttribute and removeAttribute from the interface to modify the selected attribute.

Which part of the API is public, and which is subject to change?

To answer this question, you must understand the terms used for application program interface (API) status:

- **Public** - The typical client developer codes for this API. Any severe problem in this API is addressed. Also, the majority of this API is defined as "public" and has reached World Wide Web Consortium (W3C) Recommendation status or a similar status in the XML Core Working Group. Not much change in these interfaces is expected. A W3C Recommendation indicates that a specification is stable, contributes to Web interoperability, and has been reviewed by all W3C members, who are in favor of supporting its adoption by the industry.

- **Experimental** - These interfaces and classes reflect the latest W3C specifications and Simple API for XML (SAX) specifications from the XML Core Working Group. Because these specifications are not finalized, the interfaces are subject to change. As an experimental specification is adopted at the highest level, such as at the W3C Recommendation level, the specification is upgraded to the Public category.
- **Internal** - These classes are considered internal to Xerces, even though they might be public and have public methods. Developers with complex and specific needs, such as building an XML parser can use these classes. However, the architecture is subject to change.

The following table summarizes API status by interface content:

API Status	API contents (packages, interfaces, classes and methods)	Comments
Public	DOM Level 1 interfaces (found in org.w3c.dom, org.w3c.dom.html) and DOM Level 2 (DOM2) interfaces (found in org.w3c.dom and subpackages) The DOM2 interfaces are implemented in the same interfaces as DOM1 as new methods. SAX Level 1 interfaces (found in org.xml.sax. and subpackages) and SAX Level 2 (SAX2) interfaces (found in org.xml.sax. and subpackages)	DOM L1, DOM L2, SAX1, and SAX2 Interfaces are stable.
Experimental	DOM Level 3 (DOM3) and Core DOM Level 3 (DOM3) Abstract Schemas and Load and Save (found in org.apache.xerces.dom3 and subpackages).	DOM L3 is in working draft status. XML4J provides a subset of DOM L3 support.
Internal	All other packages are internal.	The internal Xerces architecture can change.

For more information see XML4J Information.

As an Apache Open Source project, the Xerces community is interested in your questions and feedback regarding the entire API, not just the part that is designated as public.

If you have specific questions, patches, or feedback regarding the Xerces API or code, visit the Apache Web site, and sign up for the mailing list. Send more basic questions or questions specific to XML4J to the AlphaWorks discussion forum.

Which APIs do I use for new development?

Use the org.apache.xerces.parsers.* classes for new development. The four compatibility parser classes are replaced by the following classes:

```
org.apache.xerces.parsers.SAXParser
org.apache.xerces.parsers.DOMParser
```

Validation control has become a feature of the parser, and no longer requires separate classes.

The future direction for the parser instantiation classes is a parser instantiation API that results from the W3C DOM Level 3 effort, which is just getting underway.

To make sure that your code is as stable as possible, use the interfaces that are specified in the Public section of the preceding table (for example DOM1 and SAX1). Updates are made to the table to reflect the new interfaces and the classes that are given Public status (for example, DOM2 and SAX2).

Is IBM making any additional support guarantees with XML4J?

IBM is not making any additional support guarantees for XML4J. In particular, IBM is not certifying that XML4J is Y2K compliant. XML4J makes no internal date calculations.

Migrating applications to use the XML4J 4.2.2 parser and the XSLT4J 2.5.4 transformer

The Java API for XML Processing (JAXP) specification defines a pluggability mechanism for a SAX or a DOM parser using the `javax.xml.parsers` APIs. Transformers are pluggable using the `javax.xml.transform` APIs.

The IBM SDK 1.4.1 bundles in Version 6 include an XML4J 4.2.2 parser and an XSLT4J 2.5.4 transformer. You can use a different implementation of JAXP in an application. Package the parser and transformer in the application.

You can change an application to remove its dependency on the API in a previous version of the parser or the transformer from an earlier version of WebSphere Application Server. Package the JAR files in the application.

In both cases, set the class loader delegation mode to `PARENT_LAST` on the application or Web module.

Recommendation: Have your applications use the JAXP API instead of using the parser or transformer implementation API directly.

You must recompile a V4.0.x XML application or a V5.x XML application to migrate it to the Version 6 level.

Configuring WebSphere Application Server for DB2 access

This topic describes how to configure V6 on a Linux or UNIX-based platform to use multiple V8.1 DB2 clients.

WebSphere Application Server, V6 does not configure environment variables for accessing a DB2 database during installation. A database is not a prerequisite. Therefore, the installation and configuration of V6 do not involve database configuration.

You can install multiple DB2 Version 8.x instances on the same Linux or UNIX machine. This topic describes configuring V6 on a Linux or UNIX-based platform to use multiple V8.x DB2 clients.

See the IBM WebSphere Application Server supported hardware, software, and APIs Web site at <http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html> for information about required patches for DB2.

If your WebSphere Application Server, Version 6 machine has only a Version 8.x client installed, and all DB2 data sources defined in WebSphere Application Server access DB2 databases through this client, source the `db2profile` file in the login profile of your V6 instance owner, invoke the script `db2_home/java12/usejdbc2` to use the JDBC2 drivers instead of the default JDBC1 drivers, and put the `DB2 lib` directory in the `java.library.path` variable.

For example, assume that the following values are true:

WebSphere Application Server instance owner, that is the administrative user who starts WebSphere Application Server	adm00001
DB2 client instance owner	db2inst1
DB2 client instance home	/export/home/db2inst1

1. Update the `.profile` of user `adm0001`.

Add the following line:

```
. /export/home/db2inst1/sqllib/db2profile
```

You can also add the line to the `setupCmdLine.sh` script of WebSphere Application Server.

2. Specify the JDBC provider class path.

You have two ways to specify the JDBC provider class path on the DB2 JDBC Provider definition panel of the WebSphere Application Server administrative console:

- Leave the default value (`${DB2_JDBC_DRIVER_PATH}/db2java.zip`) as is. Click **Environment > Manage WebSphere Variables** in the administrative console. Set the `DB2_JDBC_DRIVER_PATH` variable to the value `/export/home/db2inst1/sqllib/java12`. This approach is preferred.
- Enter the class path, such as `/export/home/db2inst1/sqllib/java12/db2java.zip`.

Windows platforms support only one DB2 installation. The DB2 environment variables are populated in the system environment automatically. WebSphere Application Server does not need to set these environment variables.

3. Set DB2-required environment variables for a particular Application Server.

- a. On the administrative console of WebSphere Application Server, click **Application servers > server_name > Configuration > Java and process management > Environment entries > New** to create a new entry.
- b. Create a `DB2INSTANCE` entry. Enter `DB2INSTANCE` in the **Name** field, the instance name, such as `db2v8` in the **Value** field, and click **Apply**.
- c. Create a library path entry. Create the appropriate library path:
 - **AIX:** `LIBPATH` entry
 - **Linux:** `LD_LIBRARY_PATH` entry
 - **Solaris:** `LD_LIBRARY_PATH` entry
 - **HP-UX:** `SHLIB_PATH` entry

Use the value of the DB2 native library path, such as `/export/home/db2v8/sqllib/lib` for a V8.x client, or `/export/home/db2v8B/sqllib/lib` for another V8.x client.

The `DB2 lib` directory must be on the `java.library.path` path. Otherwise the application server cannot load the `db2jdbc.so` library and cannot work with DB2. Even with the `DB2 lib` directory on the `java.library.path` path, you must invoke the `/home/db2inst1/db2profile` environment into the root shell before you start the application server.

4. Configure the WebSphere Application Server product so that it can use both V8.1 clients.

In cases where multiple V8.1 clients are installed on a WebSphere Application Server V5.x machine, and you intend to use two or more clients in your V6 Application Servers, you can set DB2 environment variables within each application server, instead of setting them globally as shown previously.

DB2 UDB V8 uses a new client and server communications mechanism that is based on distributed relational database architecture. While the new communications mechanism provides a number of advantages, it also introduces restrictions on the communications capability between DB2 V7 and V8 products.

Because a number of restrictions exist when using a V8 client to communicate with a V7 server, this configuration is not recommended. However, a V7 client can access a V8 server without difficulty.

If you have a WebSphere Application Server application that accesses both a V7 server (on a WebSphere Application Server V5.x machine) and a V8 server (on a WebSphere Application Server V6 machine), use the DB2 V7 client on the WebSphere Application Server V5.x machine to access both the V7 server and the V8 DB2 server.

WebSphere Application Server V6 supports DB2 V8.x but does not support V7.

You can choose to use a V7 client to access a V7 server, and use a V8 client to access a V8 server. This choice results in two versions of DB2 clients.

When you use a V7 client (on a WebSphere Application Server V5.x machine) to access a V8 server, you must explicitly bind V7 packages to the V8 server. To bind packages in the V7 client environment, make a connection to the V8 server and bind both the `db2cli.lst` and the `db2ubind.lst` files. For example:

```
cd /home/db2inst1/sqllib/bnd
db2 connect to v8server
db2 bind @db2cli.lst
db2 bind @db2ubind.lst
db2 terminate
```

One WebSphere Application Server node can support multiple application server instances. Each application server is essentially a single Java runtime environment, which is one Java virtual machine (JVM). Each JVM can have its own set of environment variables that differ from other application server instances.

You can set DB2 environment variables per application server instance to let each application server instance communicate with a single DB2 client instance. The client instance can have multiple databases cataloged.

Such a configuration means that you cannot have one application server instance communicating with both a V7 client and a V8.1 client. However, you can create another application server instance to communicate with a different DB2 client.

5. Match the appropriate data source to the application component.

When mapping a data source to an application component, do not mismatch data sources from different DB2 client instances.

For example, if you set the `server1` application server instance to run in the DB2 V7 client instance, `server1` application components can use only data sources defined under the same V7 client JDBC driver.

WebSphere Application Server V6 supports defining a JDBC provider on different scopes: cell, cluster, node (the default) and server. If you have different DB2 client instances, consider defining them on the server level instead of on the node or cluster level. Server level definitions avoid possible mismatches between the data sources and different DB2 JDBC providers.

You can configure DB2 for use on WebSphere Application Server. Tune your WebSphere Application Server environment as you test it. Test all redeployed applications before moving them into production.

Chapter 6. Migrating Web server configurations

This topic describes what to do to migrate a Web server from supporting IBM WebSphere Application Server, Version 4.0.x or V5.x, to support IBM WebSphere Application Server- Express, Version 6.

Use the Plugins installation wizard to install the binary plug-in module and the `plugin-cfg.xml` configuration file on the same machine as the Web server. Use the following procedure to migrate your Web server to work with Version 6.

1. Install the IBM HTTP Server, Version 6.0 and its plug-in, or a plug-in for another supported Web server from the product CD-ROM.

Install the HTTP Server and its plug-in on a different machine with the following procedure:

- a. Insert the product CD-ROM labelled, **WebSphere Application Server, IBM HTTP Server**, into the machine.
- b. Close the launchpad if it starts automatically.
- c. Change directories to the IHS directory on the product CD-ROM.
- d. Click the installation script for your platform, to install the IBM HTTP Server:

-   **InstallIHS.sh**
-  **InstallIHS.bat**

This script installs the plug-in you need and makes the necessary configuration changes for the supported Web server.

IBM HTTP Server, Version 6.0 can coexist with earlier versions, or you can upgrade earlier versions to V6.0. Upgrading relieves you from having to uninstall and reinstall the HTTP server. Install V6.0 into the same directory structure as the earlier version to upgrade that version. If you install the HTTP Server into a different directory, V6.0 coexists with the previous version. By default, the administration server and the Web Server use the same ports as the previous version, which causes a conflict. However, you can change the port assignments on the port assignment panel of the Installation wizard or the Profile creation wizard.

- Change the port number assignments for the new installation if you install into a separate directory. You can change port numbers on the coexistence panel. You can back track through the installation wizard and change the port settings if you have not already done so. Or, you can change the port settings after installation, in the `httpd.conf` file in the HTTP Server directory.
- Update the IBM HTTP Server `httpd.conf` configuration entries to remove entries for earlier WebSphere Application Server versions if you install into the same directory as an earlier version. Versions 4.0.x, 5.x, and 6 of WebSphere Application Server use the same HTTP transport plug-in binary module. If the Web server configuration file contains WebSphere Application Server V4.0.x or V5.x plug-in information, you must manually remove it. Otherwise, when the HTTP Server attempts to start the second V6 plug-in binary module, there is an error. The error indicates that the module is already loaded.

The configuration file might contain duplicate entries for accessing WebSphere Application Server Samples. Remove any aliases for previous versions and retain the V6 entries:

V4.0.x installations:

```
Alias /IBMWebAS/ "C:\WebSphere\AppServer40\web\  
Alias /WSsamples "C:\WebSphere\AppServer40\WSsamples"
```

V5 installation:

```
Alias /WSsamples "c:\Program Files\WebSphere\AppServer\WSsamples"  
Alias /IBMWebAS/ "c:\Program Files\WebSphere\AppServer\web"
```

V6 installation:

```
Alias /WSsamples "c:\Program Files\WebSphere\AppServer\WSsamples"  
Alias /IBMWebAS/ "c:\Program Files\WebSphere\AppServer\web"
```

2. Migrate plug-ins to work with IBM WebSphere Application Server, Version 6. Starting with WebSphere Application Server V6.0:

- Web servers can be represented in the administrative console.
- The Web server plug-in configuration file (`plugin-cfg.xml`) is associated with every Web server definition instead of one cell-wide plug-in configuration file.
- The settings in the generated plug-in configuration file are based on the list of applications that are deployed on the hosting Web server.

Use the following steps to generate a Web server plug-in configuration file that is based on topology. This method was used for generated plug-in configuration files in previous releases.

- a. Use the `GenPluginCfg.bat` or `GenPluginCfg.sh` script to generate the plug-in configuration file.
- b. Manually propagate the generated plug-in configuration file from the machine on which the WebSphere Application Server resides to the remote Web server.
- c. Use the Plug-ins installation wizard to configure the Web server. Instead of using the default plug-in configuration file location, specify the new location of the plug-in configuration file that was propagated in the previous step.

It is recommended that you migrate to the application-centric approach which uses the Plug-ins installation wizard. The Plug-ins installation wizard generates scripts that can be used to create the Web server definition for that Web server and to map all of the applications that are currently deployed to the newly created Web server definition.

You now understand what you must do to install or migrate a Web server and plug-in to work with WebSphere Application Server, including installing the Web server on the same machine with the Application Server, or on different machines.

After migrating your Web server, you can install the WebSphere Application Server product or migrate a previous installation:

Chapter 7. Migrating administrative scripts

WebSphere Application Server Version 6 supports migrating administrative scripts from V4.0.x and V5.x.

- If you are migrating administrative scripts from V4.0.x, see:
“Migrating V4.0.x administrative scripts to V6 wsadmin”
- If you are migrating administrative scripts from V5.x, see:
“Migrating administrative scripts from 5.x to 6.0” on page 95

Migrating V4.0.x administrative scripts to V6 wsadmin

The purpose of this section is to provide guidance for migrating from WebSphere Application Server V4.0 wscp scripts to wsadmin in V6.x.

The wscp tool was a part of the WebSphere Application Server V4.0 administration repository support. The repository no longer exists and the tools that manipulate it are no longer needed. You can use the V6.x scripting client program, wsadmin, to do the same kinds of things wscp did, and more. You can use the Jacl and Jython scripting languages for scripts, but the elements specific to wsadmin are different from those available in wscp. This article shows how to create WebSphere Application Server V6.x scripts that perform actions similar to those performed by V4.0 wscp. Automatic conversion of scripts between the two releases is difficult.

The wsadmin scripting client uses the Bean Scripting Framework (BSF), and is based on Java Management Extensions (JMX).

In V4.0, wscp commands are used for both configuration queries or updates, and operational commands. In V6.x, a distinction is made between configurational and operational commands.

1. Identify the wscp commands used in your script.
2. Determine if each command is used for configuration or operation.
 - Configuration commands include the following: create, list, modify, remove, show, showall, install, uninstall, all SecurityConfig commands, all SecurityRoleAssignment commands, clone, and removeClone.
 - Operation commands include the following: start, stop, show (if for a run-time attribute), testConnection, all DrAdmin commands, and regenPluginCfg.
 - Other commands exist to provide help for configuration commands. These commands include the following: attributes, containment, and help.
3. Find the corresponding configuration wsadmin V6.x object type for each configuration commands. Use the AdminConfig create, list, modify, remove, show, and showAttribute commands to perform the same type of operations in V6.x that you performed in V4.0. Use the following table to determine the corresponding types:

V4.0 wscp command	V6.x wsadmin configuration type
ApplicationServer	Server
Context	Not applicable
DataSource	WAS40DataSource, DataSource
Domain	Not applicable
EnterpriseApp	ApplicationDeployment
GenericServer	Server
J2CConnectionFactory	J2CConnectionFactory
J2CResourceAdapter	J2CResourceAdapter

JDBCProvider	JDBCProvider
JMSConnectionFactory	JMSConnectionFactory
JMSDestination	JMSDestination
JMSProvider	JMSProvider
MailSession	MailSession
Module	ModuleDeployment
Node	Node
ServerGroup	ServerCluster
URL	URL
URLProvider	URLProvider
VirtualHost	VirtualHost

- Determine the V6.x attribute names by using the online help commands of the AdminConfig object. For example: attributes, defaults, parents, required, or types.
- Convert application installation commands. For application installation, use the AdminApp object **installInteractive** command to complete a successful installation. Then locate message WASX7278I in the wsadmin.traceout log file and use the data in the message to construct an installation command for your source.
- Convert operational commands. Use the following table to determine how to deal with operational commands in V6.x wscp:

wscp 4.0	wsadmin 6.x	wsadmin 6.x	wsadmin 6.x
action	Object and command	Mbean, if any	Operation, if any
server start	AdminControl startServer		
server stop	AdminControl stopServer		
servergroup start	AdminControl invoke	Cluster	start
servergroup stop	AdminControl invoke	Cluster	stop
application start	AdminControl invoke	ApplicationManager	startApplication
application stop	AdminControl invoke	ApplicationManager	stopApplication
node stop	AdminControl invoke	<nodeagent>	stopNode
check run-time attribute	AdminControl getAttribute	<mbean>	<attribute>
check run-time attributes	AdminControl getAttributes	<mbean>	<list of attributes>
regenPluginCfg	AdminControl invoke	PluginCfgGenerator	generate
testConnection	AdminControl testConnection		
enable security	securityon command		
disable security	securityoff command		

- Save configuration changes. In V6.x, configuration changes are made to a temporary workspace. These changes are not committed to the WebSphere Application Server configuration until you invoke the **save** command on the AdminConfig object. If your script makes configuration changes, for example, creates, removes, or changes objects, or installs or uninstalls applications, invoke the following command to commit the change:

Using Jacl:

```
$AdminConfig save
```

Using Jython:

```
AdminConfig.save()
```

Example: Migrating - Creating an application server

Creating an application server involves a configuration command. To do this task in the wscp V4.0 tool and the wsadmin V6.x tool, you must know the hierarchical name of the application server. The following examples demonstrate how to create an application server in WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
ApplicationServer create /Node:mynode/ApplicationServer:myserv/ -attribute {{Stdout myfile.out}}
```

- wsadmin V6.x

Server objects are contained within nodes.

Using Jacl:

```
set node [$AdminConfig getid /Node:mynode/]
$AdminConfig create Server $node {{name myserv} {outputStreamRedirect {{fileName myfile.out}}}}
$AdminConfig save
```

Using Jython:

```
node = AdminConfig.getid('/Node:mynode/')
AdminConfig.create('Server', node, [['name', 'myserv'],
    ['outputStreamRedirect', [['fileName', 'myfile.out']]])
AdminConfig.save()
```

where Stdout is the name of the V4.0 ApplicationServer attribute that is replaced by the fileName attribute, and embedded within the outputStreamRedirect attribute of the server.

Example: Migrating - Starting an application server

The following examples demonstrate how to start an application server with WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
ApplicationServer start /Node:mynode/ApplicationServer:myserv/
```

- wsadmin V6.x

If you are connected to a server in a base installation, you cannot request to start another server. You can only start an application server if you have a Network Deployment installation. In a Network Deployment installation, use the following command:

Using Jacl:

```
$AdminControl startServer myserv mynode 600
```

Using Jython:

```
AdminControl.startServer('myserv', 'mynode', 600)
```

where 600 represents the wait time in seconds. The server name and node name are required.

Example: Migrating - Installing an application

The following examples demonstrate how to install an application in WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

– Construct the -modvirtualhosts option:

```
set modhost1 [list mtcomps.war default_host]
set modhosts [list $modhost1]
```

– Construct the -resourcereferences option:

```
set resref1 [list mtcomps.war::mail/MailSession9 mail/DefaultMailSession]
set resref2 [list deplmtest.jar::MailEJBObject::mail/MailSession9 mail/DefaultMailSession]
set resrefs [list $resref1 $resref2]
```

– Install the application:

```
EnterpriseApp install /Node:mynode/ c:/WebSphere/AppServer/installableApps/jmsample.ear
  -appname MailSampleApp
  -defappserver /Node:$mynode/ApplicationServer:myserv/
  -modvirtualhosts $modhosts
  -resourcereferences $resrefs
```

- wsadmin V6.x

The following command sequence accomplishes approximately the same thing as the V4.0 commands previously presented, but simpler ways exist:

- Construct the `-MapWebModToVH` option:

Using Jacl:

```
set modtovh1 [list "JavaMail Sample WebApp" mtcomps.war,WEB-INF/web.xml default_host]
set modtovh [list $modtovh1]
```

Using Jython:

```
modtovh1 = ["JavaMail Sample WebApp", 'mtcomps.war,WEB-INF/web.xml', 'default_host']
modtovh = [modtovh1]
```

- Construct the `-MapResRefToEJB` option:

Using Jacl:

```
set resreftoejb1 [list deplmtest.jar MailEJBObject
  deplmtest.jar,META-INF/ejb-jar.xml
  mail/MailSession9
  javax.mail.Session
  mail/DefaultMailSession]
set resreftoejb2 [list "JavaMail Sample WebApp" ""
  mtcomps.war,WEB-INF/web.xml
  mail/MailSession9
  javax.mail.Session mail/bozo]
set resreftoejb [list $resreftoejb1 $resreftoejb2]
```

Using Jython:

```
resreftoejb1 = [['deplmtest.jar', 'MailEJBObject',
  'deplmtest.jar,META-INF/ejb-jar.xml',
  'mail/MailSession9',
  'javax.mail.Session',
  'mail/DefaultMailSession']]
resreftoejb2 = ["JavaMail Sample WebApp", ' ',
  'mtcomps.war,WEB-INF/web.xml',
  'mail/MailSession9',
  'javax.mail.Session',
  'mail/bozo']]
```

- Construct the attribute string:

Using Jacl:

```
set attrs [list -MapWebModToVH $modtovh
  -MapResRefToEJB $resreftoejb
  -node mynode
  -server myserv
  -appname MailSampleApp]
```

Using Jython:

```
attrs = ['-MapWebModToVH', modtovh,
  '-MapResRefToEJB', resreftoejb1,
  '-MapResRefToEJB', resreftoejb2,
  '-node', 'mynode',
  '-server', 'myserv',
  '-appname', 'MailSampleApp']
```

- Install the application:

Using Jacl:

```
$AdminApp install c:/WebSphere/AppServer/installableApps/jmsample.ear $attrs
```

Using Jython:

```
AdminApp.install('c:/WebSphere/AppServer/installableApps/jmsample.ear', attrs)
```

- Save your changes:

```
$AdminConfig save
```

Using Jython:

```
AdminConfig.save()
```

You can use the AdminApp **taskInfo** command to obtain information about each task option. You can use the AdminApp **interactiveInstall** command to step through all the installation tasks, one at a time. If you use the **installInteractive** command to successfully install an application, an option string logs in the wsadmin.traceout file under the message ID WASX7278I. You can copy and paste this option string into wsadmin scripts.

Example: Migrating - Installing a JDBC driver

The following examples demonstrate how to install a JDBC driver in WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

In the WebSphere Application Server V4.0, you must create the JDBC driver and then install it.

```
JDBCdriver create /JDBCdriver:mydriver/  
-attribute {{ImplClass COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource}}  
JDBCdriver install /JDBCdriver:mydriver/  
-node /Node:mynode/ -jarFile c:/SQLLIB/java/db2java.zip
```

- wsadmin V6.x

In WebSphere Application Server V6.x, there is no separate installation step. The Java archive (JAR) file name in the V4.0 installation step is replaced by the classpath attribute on the V6.x JDBC provider object. In V6.x, resources can exist at the server, node, or cell level of the configuration.

Using Jacl:

```
set node [$AdminConfig getid /Node:mynode/]  
$AdminConfig create JDBCProvider $node  
  {{classpath c:/SQLLIB/java/db2java.zip}  
  {implementationClassName COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource}  
  {name mydriver}}  
$AdminConfig save
```

Using Jython:

```
node = AdminConfig.getid('/Node:mynode/')  
AdminConfig.create('JDBCProvider', node,  
  [['classpath', 'c:/SQLLIB/java/db2java.zip'],  
  ['implementationClassName',  
   'COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource'],  
  ['name', 'mydriver']])  
AdminConfig.save()
```

Example: Migrating - Creating a server group

In WebSphere Application Server V6.x, ServerClusters have replaced V4.0 ServerGroups. The members of a cluster are servers with identical application configurations. The following examples demonstrate how to create a server group in WebSphere Application Server V4.0 and V6.x. They assume that an application server named *as1* already exists and is used as the first clone in a server group:

- wscp V4.0

```
ServerGroup create /ServerGroup:sg1/ -baseInstance /Node:mynode/ApplicationServer:as1/  
-serverGroupAttrs {{EJBServerAttributes {{SelectionPolicy roundrobin}}}}
```

- wsadmin V6.x

Using Jacl:

```
set serverid [$AdminConfig getid /Node:mynode/Server:as1/]  
$AdminConfig convertToCluster $serverid MyCluster  
$AdminConfig save
```

Using Jython:

```
serverid = AdminConfig.getid('/Node:mynode/Server:as1/')  
AdminConfig.convertToCluster(serverid, 'MyCluster')  
AdminConfig.save()
```


Example: Migrating - Stopping a node

The WebSphere Application Server V4.0 wscp tool requires the name of the node. In V4.0, if you stop a node, you bring down the administrative server on that node. To take the equivalent action in V6.x, stop the node agent.

Tip: If you bring down the server to which the wsadmin process is connected, you cannot issue any more commands against that server.

- wscp V4.0

```
Node stop /Node:mynode/
```

- wsadmin V6.x

Using Jacl:

```
set na [$AdminControl queryNames type=NodeAgent,node=mynode,*]
$AdminControl invoke $na stopNode
```

Using Jython:

```
na = AdminControl.queryNames('type=NodeAgent,node=mynode,*')
AdminControl.invoke(na, 'stopNode')
```

Stopping the node agent on a remote machine process is an asynchronous action where the stop is initiated, and then control returns to the command line.

Example: Migrating - Stopping an application server

The WebSphere Application Server V4.0 wscp tool requires that you know the hierarchical name of the application server in question (the node name and the server name). The same information is required in V6.x.

Tip: You are stopping a server object, not an application server. Servers represent logical processes on many platforms, for instance Windows or AIX, and are the entity that is stopped and started. Application servers are contained within servers.

- wscp V4.0

```
ApplicationServer stop {/Node:mynode/ApplicationServer:Default Server/}
```

- wsadmin V6.x

Using Jacl:

```
$AdminControl stopServer servername [nodename immediateFlag]
```

Using Jython:

```
AdminControl.stopServer('servername', [nodename immediateFlag])
```

For a Network Deployment installation:

Using Jacl:

```
$AdminControl stopServer servername nodename [immediateFlag]
```

Using Jython:

```
AdminControl.stopServer('servername', 'nodeName', [immediateFlag])
```

Example: Migrating - Pinging running servers for the current state

The purpose of this task is to determine if a server is running. The following examples demonstrate how to ping running servers in WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
set servers [ApplicationServer list]
foreach server $servers {
    set result ApplicationServer show $server -attribute {CurrentState}
    puts "state for server $server: $result"
}
```

- wsadmin V6.x

In the WebSphere Application Server V6.x configuration and control commands are separate.

Using Jacl:

```
set servers [$AdminConfig list Server]
foreach server $servers {
  set objName [$AdminConfig getObjectName $server]
  if {[length $objName] == 0} {
    puts "server $server is not running"
  } else {
    set result [$AdminControl getAttribute $objName state]
    puts "state for server $server: $result"
  }
}
```

Using Jython:

```
# get line separator
import java.lang.System as sys
lineSeparator = sys.getProperty('line.separator')

servers = AdminConfig.list('Server').split(lineSeparator)
for server in servers:
  objName = AdminConfig.getObjectName(server).split(lineSeparator)
  if len(objName) == 0:
    print "server " + server + " is not running\n"
  else:
    result = AdminControl.getAttribute(objName, 'state')
    print "state for server " + server + ": " + result + "\n"
```

The first line of this example obtains a list of all servers defined in the configuration. You can interrogate this data to determine the running servers. If the server is not running, nothing is returned from the getObjectName command on the AdminConfig object. If the server is running, ask for its state attribute. If the Mbean is there, the server is running and the state is STARTED. It is possible, however, for the state to be something other than STARTED, for example, STOPPING.

Example: Migrating - Regenerating the node plug-in configuration

The following examples demonstrate how to regenerating the node plug-in configuration in the WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
Node regenPluginCfg /Node:mynode/
```

- wsadmin V6.x

Using Jacl:

```
set generator [$AdminControl completeObjectName type=PluginCfgGenerator,node=mynode,*]
$AdminControl invoke $generator generate "c:/WebSphere/DeploymentManager
c:/WebSphere/DeploymentManager/config mycell mynode null plugin-cfg.xml"
```

Using Jython:

```
generator = AdminControl.completeObjectName('type=PluginCfgGenerator,node=mynode,*')
AdminControl.invoke(generator, 'generate', "c:/WebSphere/DeploymentManager
c:/WebSphere/DeploymentManager/config mycell mynode null plugin-cfg.xml")
```

Example: Migrating - Testing the DataSource object connection

The following examples demonstrate how to test the connection to a DataSource object in the WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
set myds /JDBCdriver:mydriver/DataSource:myds/
DataSource testConnection $myds
```

- wsadmin V6.x

The **testConnection** command is part of the AdminControl object because it is an operational command. This particular type of operational command takes a configuration ID as an argument, so you invoke the **getid** command on the AdminConfig object:

Using Jacl:

```
set myds [$AdminConfig getid /JDBCProvider:mydriver/DataSource:mydatasrc/]
$AdminControl testConnection $myds
```

Using Jython:

```
myds = AdminConfig.getid('/JDBCProvider:mydriver/DataSource:mydatasrc/')
AdminControl.testConnection(myds)
```

In many cases, a user ID and password, or other properties are required to complete the test connection. If this is the case, you receive the following message, which describes the missing properties:

```
WASX7216E: 2 attributes required for testConnection are missing:
"[user, password]" To complete this operation,
please supply the missing attributes as an option,
following this example:
{{user user_val} {password password_val}}
```

For this example, issue the following commands:

Using Jacl:

```
set myds [$AdminConfig getid /JDBCProvider:mydriver/DataSource:mydatasrc/]
$AdminControl testConnection $myds {{user myuser} {password secret}}
```

Using Jython:

```
myds = AdminConfig.getid('/JDBCProvider:mydriver/DataSource:mydatasrc/')
AdminControl.testConnection(myds, [['user', 'myuser'], ['password', 'secret']])
```

Example: Migrating - Cloning a server group

The following examples demonstrate how to clone a server group in WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
ServerGroup clone /ServerGroup:sg1/ -cloneAttrs {{Name newServer}} -node /Node:mynode/
```

- wsadmin V6.x

In the following example, the first command obtains the cluster ID, the second command obtains the node ID, and the last command creates a new member of an existing cluster:

Using Jacl:

```
set cluster1 [$AdminConfig getid /ServerCluster:mycluster/]
set n1 [$AdminConfig getid /Node:mynode/]
$AdminConfig createClusterMember $cluster1 $n1 {{memberName newServer}}
$AdminConfig save
```

Using Jython:

```
cluster1 = AdminConfig.getid('/ServerCluster:mycluster/')
n1 = AdminConfig.getid('/Node:mynode/')
AdminConfig.createClusterMember(cluster1, n1, [['memberName', 'newServer']])
AdminConfig.save()
```

Example: Migrating - Enabling security

The following examples demonstrate how to enable local OS security for WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
SecurityConfig setAuthenticationMechanism LOCALOS -userid {me secret}
SecurityConfig enableSecurity
```

- wsadmin V6.x

Using Jacl:

```
securityon myuser mypassword
```

Using Jython:

```
securityon('myuser', 'mypassword')
```

This command turns on local security. The securityon function checks the validity of the user and password combination, and fails the function if the combination is invalid.

Note: This action assumes that global security is fully configured before issuing this command and that you are just switching the enabled flag on and off. If global security is not yet fully configured, the command fails.

Example: Migrating - Disabling security

The following examples demonstrate how to disable security for WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
SecurityConfig disableSecurity
```

- wsadmin V6.x

Using Jacl:

```
securityoff
```

Using Jython:

```
securityoff()
```

This command turns off global security.

Example: Migrating - Modifying the virtual host

The following examples demonstrate how to modify the virtual host in WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
VirtualHost modify /VirtualHost:default_host/ -attribute {{Name default_host} {AliasList  
{*:80 *:9080 *:9081}}}
```

- wsadmin V6.x

Using Jacl:

```
set def_host [$AdminConfig getid /VirtualHost:default_host/]  
$AdminConfig modify $def_host  
  {{aliases {{{port 80}  
    {hostname *}  
    {{port 9080}  
      {hostname *}  
    {{port 9081}  
      {hostname *}}}}}}
```

```
$AdminConfig save
```

Using Jython:

```
def_host = AdminConfig.getid('/VirtualHost:default_host/')  
AdminConfig.modify(def_host,  
  [['aliases',  
    [[['port', 80],  
      ['hostname', '*']], [['port', 9080],  
      ['hostname', '*']], [['port', 9081],  
      ['hostname', '*']]]]])  
AdminConfig.save()
```

Example: Migrating - Modifying and restarting an application server

In this task, you make a configuration change to an existing application server, then stop and restart the server to pick up the change. In WebSphere Application Server V6.x, you can only change the attributes in a running server that the server supports explicitly, or by objects it contains. You can determine these attributes online by using the `Help attributes scripting` command, or by referring to the Mbean documentation. When you use this type of update, you change only the current running state of the server. Your changes are not permanent. The updates that you make to the server configuration do not take effect until you stop and restart the server.

The enum attribute is changed. WebSphere Application Server V4.0 requires that you find the corresponding integer to enum value by making changes with the GUI, and examining the output. In V6.x, the string names of the enum literals are available using online help, using the AdminConfig **attributes** command, and displaying or updating an attribute.

The following examples demonstrate how to modify and restart an application server in WebSphere Application Server V4.0 and V6.x:

- wscp V4.0
 1. Stop the application server using the following command:

```
ApplicationServer stop /Node:mynode/ApplicationServer:myserver/
```
 2. Modify the application server, for example:

```
ApplicationServer modify /Node:mynode/ApplicationServer:myserver/ -attribute {{ModuleVisibility 1}}
```
 3. Restart the application server using the following command:

```
ApplicationServer start /Node:mynode/ApplicationServer:myserver/
```
- wsadmin V6.x
 1. Stop the application server using the following command:

Using Jacl:

```
$AdminControl stopServer myserver mynode
```

Using Jython:

```
AdminControl.stopServer('myserver', 'mynode')
```
 2. Modify the application server, for example:

Using Jacl:

```
set s1 [$AdminConfig getid /Node:mynode/Server:myserver/]
set errStream [$AdminConfig showAttribute $s1 errorStreamRedirect]
$AdminConfig modify $s1 {{rolloverPeriod 12}}
$AdminConfig save
```

Using Jython:

```
s1 = AdminConfig.getid('/Node:mynode/Server:myserver/')
errStream = AdminConfig.showAttribute(s1, 'errorStreamRedirect')
AdminConfig.modify(s1, [['rolloverPeriod', 12]])
AdminConfig.save()
```
 3. Restart the application server using the following command:

Using Jacl:

```
$AdminControl startServer myserver mynode
```

Using Jython:

```
AdminControl.startServer('myserver', 'mynode')
```

Example: Migrating - Removing an application server

Removing an application server involves a configuration command:

- wscp V4.0

```
ApplicationServer remove /Node:mynode/ApplicationServer:myserv/
```
- wsadmin V6.x

Using Jacl:

```
set serv [$AdminConfig getid /Node:mynode/Server:myserv/]
$AdminConfig remove $serv
$AdminConfig save
```

Using Jython:

```
serv = AdminConfig.getid('/Node:mynode/Server:myserv/')
AdminConfig.remove(serv)
AdminConfig.save()
```

Example: Migrating - Modifying the embedded transports in an application server

The following examples demonstrate how to modify the embedded transports in an application server in WebSphere Application Server V4.0 and V6.x:

- wscp V4.0

```
ApplicationServer modify /Node:mynode/ApplicationServer:myserv/ -attribute
  {{WebContainerConfig {Transports {{{{MaxKeepAlive 25} {MaxReqKeepAlive 100} {KeepAliveTimeout
  5} {ConnectionTimeout 5} {Host *} {Port 9080} {BacklogConnections 511} {HttpProperties {}}
  {SSLEnabled true} {SSLConfig {}}}}}}}}
```

- wsadmin V6.x

Using Jacl:

```
set server [$AdminConfig getid /Node:mynode/Server:myserv/]
set web_container [$AdminConfig list WebContainer $server]
$AdminConfig modify $web_container
  {{transports:HTTPTransport
  {{sslEnabled true}
  {sslConfig DefaultSSLSettings}
  {address {{host *} {port 9080}}}}}}
$AdminConfig save
```

Using Jython:

```
server = AdminConfig.getid('/Node:mynode/Server:myserv/')
web_container = AdminConfig.list('WebContainer', server)
AdminConfig.modify(web_container,
  [['transport',
  [['sslEnabled', 'true'],
  ['sslConfig', 'DefaultSSLSettings'],
  ['address', [['host', '*'],
  ['port', 9080]]]]]])
AdminConfig.save()
```

Example: Migrating - Connecting to a remote server

- wscp V4.0

```
wscp.hostname = myhost
```

- wsadmin V6.x

```
wsadmin -host myhost.austin.ibm.com
```

Example: Migrating - Uninstalling an application

- wscp V4.0

```
EnterpriseApp remove {/EnterpriseApp:Sample Application/}
```

- wsadmin V6.x

Using Jacl:

```
$AdminApp uninstall TechnologySamples
$AdminConfig save
```

Using Jython:

```
AdminApp.uninstall('TechnologySamples')
AdminConfig.save()
```

Example: Migrating - Editing an application

- wscp V4.0

```
EnterpriseApp modify {/EnterpriseApp:Trade Sample/} -attribute {{Name changedName}}
```

- wsadmin V6.x

Using Jacl:

```
$AdminApp edit DefaultApplication {-BindJndiForEJBMessageBinding
  {"Increment Enterprise Java Bean" "Increment"
  "Increment.jar,META-INF/ejb-jar.xml" "Increment"}}}
$AdminConfig save
```

Using Jython:

```
AdminApp.edit('DefaultApplication', '[-BindJndiForEJBMessageBinding
  [{"Increment Enterprise Java Bean" Increment
  Increment.jar,META-INF/ejb-jar.xml Increment}]]')
AdminConfig.save()
```

Example: Migrating - Modifying attributes of application servers, applications, and other configured objects

- wscp V4.0

```
ApplicationServer modify /Node:dev-pc/ApplicationServer:myServer/
-attribute {{PingInterval 120}}
```

- wsadmin V6.x

Using Jacl:

```
set trace [$AdminConfig getid /TraceService:/]
$AdminConfig modify trace {{startupTraceSpecification com.ibm.ws.*=all=enabled}}
$AdminConfig save
```

Using Jython:

```
trace = AdminConfig.getid('/TraceService/')
AdminConfig.modify('trace', [[startupTraceSpecification com.ibm.ws.*=all=enabled]])
AdminConfig.save()
```

You must restart the server in order to see the changes in the configuration.

Example: Migrating - Displaying help

- wscp V4.0

To obtain general help, use the following example:

```
help
```

To obtain help on a particular object, use the following example:

```
ApplicationServer help
```

- wsadmin V6.x

To list all of the options available, use the following example:

Using Jacl:

```
$Help scriptingObject
```

Using Jython:

```
Help.scriptingObject()
```

where *scriptingObject* can be AdminConfig, AdminControl, AdminApp or AdminTask.

Example: Migrating - Listing actions available for configured objects

- wscp V4.0

```
ApplicationServer help operations
```

- wsadmin V6.x

Using Jacl:

```
$AdminConfig attributes Server
```

Using Jython:

```
AdminConfig.attributes('Server')
```

Example: Migrating - Setting the server trace specification

- wscp V4.0
DrAdmin remote <portno> -setTrace com.ibm.ejs.*=all=enabled

- wsadmin V6.x

There are two ways to set tracing with wsadmin in V6.x. The following way takes immediate effect, but is temporary and is set on the runtime, using the AdminControl object:

Using Jacl:

```
set ts [$AdminControl queryNames type=TraceService,node=nodeName,process=serverName,*]
$AdminControl setAttribute $ts traceSpecification com.ibm.*=all=enabled
```

Using Jython:

```
ts = AdminControl.queryNames('type=TraceService,node=nodeName,process=serverName,*')
AdminControl.setAttribute(ts, 'traceSpecification', 'com.ibm.*=all=enabled')
```

If you want your changes to persist, change the configuration by using the AdminConfig object, for example:

Using Jacl:

```
set svr [$AdminConfig getid /Node:nodeName/Server:serverName/]
set ts [$AdminConfig list TraceService $svr]
$AdminConfig modify $ts {{startupTraceSpecification com.ibm.*=all=enabled}}
$AdminConfig save
```

Using Jython:

```
svr = AdminConfig.getid('/Node:nodeName/Server:serverName/')
ts = AdminConfig.list('TraceService', svr)
AdminConfig.modify(ts, [['startupTraceSpecification', 'com.ibm.*=all=enabled']])
AdminConfig.save()
```

You can also change the TraceLog specifications, for example:

Using Jacl:

```
set svr [$AdminConfig getid/Node:nodeName/Server:serverName/]
set ts [$AdminConfig list TraceService $svr]
set trlog [$AdminConfig list TraceLog $ts]
$AdminConfig modify $trlog {{fileName myFile.log} {maxNumberOfBackupFiles 10} {rolloverSize 2}}
$AdminConfig save
```

Using Jython:

```
svr = AdminConfig.getid('/Node:nodeName/Server:serverName/')
ts = AdminConfig.list('TraceService' svr)
trlog = AdminConfig.list('TraceLog', ts)
AdminConfig.modify(trlog, [['fileName' 'myFile.log'], ['maxNumberOfBackupFiles' 10], ['rolloverSize' 2]])
AdminConfig.save()
```

Migrating administrative scripts from 5.x to 6.0

There are a few changes to be aware of that are required for your existing scripts when moving to WebSphere Application Server Version 6. In general, the administration model has changed very little. However, there are some changes required with Version 6. See the following steps for the three categories of changes:

1. Prepare for evolutionary changes **without** script compatibility support. These types of changes can be evolved directly without the assistance of script compatibility support. The data can be accessed from multiple locations, including the old and new locations. As long as the new location is not updated, the data is accessed from the old location. Once the new location is updated, it becomes the current data and is used for further accesses and updates. Warning messages are logged when the old location is still being used.

The location of the transaction logs directory attribute has changed from the ApplicationServer::TransactionService to the ServerEntry::recoveryLogs. As long as the new

location is not used, the value from the old location will continue to be used. Scripts that modify the old location can still be used; that value will take effect until a value in the new location is set. The change to scripts to use the new location is as follows:

Old location:

- Using Jacl:

```
set transService [$AdminConfig list TransactionService $server1]
  $AdminConfig showAttribute $transService transactionLogDirectory
```

New Location:

- Using Jacl:

```
$AdminConfig list ServerEntry $node
set serverEntry <select one of the ServerEntry from output of above command>
set recoveryLog [$AdminConfig showAttribute $serverEntry recoveryLog]
$AdminConfig showAttribute $recoveryLog transactionLogDirectory
```

2. Prepare for evolutionary changes **with** script compatibility support. These changes are assisted by the compatibility mode provided by “WASPostUpgrade command” on page 140. During migration, the default is to migrate using compatibility mode. If this option is taken, then the old object types are migrated into the new configuration; all existing scripts will run unchanged.

HTTP transports: the new architecture for V6 uses the new channel framework. HTTP definitions are mapped on top of this support. When compatibility mode is chosen, the old HTTPTransport objects are migrated and mapped onto the channel architecture. Existing scripts can modify these objects and will run unchanged. See “Example: Migrating - Modifying the embedded transports in an application server” on page 93 for examples.

Process definition: The name of this object is changed from processDef to processDefs. You can mitigate this change by using the compatibility mode mapping provided by the migration tools. The change to scripts to use the new location is as follows:

Old example:

- Using Jacl:

```
set processDef [$AdminConfig list JavaProcessDef $server1]
set processDef [$AdminConfig showAttribute $server1 processDefinition]
```

Using Jython:

```
processDef = AdminConfig.list('JavaProcessDef', server1)
print processDef
```

New example. Identify the process definition belonging to this server and assign it to the processDefs variable:

- Using Jacl:

```
set processDefs [$AdminConfig list JavaProcessDef $server1]
set processDefs [$AdminConfig showAttribute $server1 processDefinitions]
```

Using Jython:

```
processDefs = AdminConfig.list('JavaProcessDef', server1)
print processDefs
```

3. Prepare for required changes.
 - a. Be aware of the implications of migrating JMS applications from the embedded messaging in WebSphere Application Server Version 5 to the default messaging provider in WebSphere Application Server Version 6.
 - b. A new version of Jacl (1.3.1) is shipped with WebSphere Application Server V6. With this Jacl version, regexp command supports only tcl 8.0 regexp command syntax. If your existing V5.x Jacl script uses regexp command syntax that is supported in Jacl 1.2.6 but not anymore in Jacl 1.3.1, you may not get a match anymore, or you may get a compile error for your regexp command similar to the following:

```
com.ibm.bsf.BSFException: error while eval'ing Jacl expression:
couldn't compile regular expression pattern: ?+* follows nothing
while executing
"regexp {(?x)
```



```

... "
  ("if" test expression)
  invoked from within
"if {[regex {(?x)}
... "
  (file testregexp.jacl line 2)
  (file line 2)
  invoked from within
"source testregexp.jacl"

```

There is no workaround for this regression problem. Jacl has indicated that this is by design and there is no simple patch to fix this design decision. For more information on Jacl, see the *Administration: Operations* PDF.

Example: Migrating - Allowing configuration overwrite when saving a configuration

The following examples demonstrate how to enable configuration overwrite in network deployment for WebSphere Application Server V5.x and V6.x:

- wsadmin V5.x

Using Jacl:

```
$AdminConfig setSaveMode overwriteOnConflict
```

Using Jython:

```
AdminConfig.setSaveMode('overwriteOnConflict')
```

- wsadmin V6.x

1. Enable configuration repository to allow configuration overwrite:

Using Jacl:

```

set s1AdminService [$AdminConfig getid /Server:dmgr/AdminService:/]

set configRepository [$AdminConfig showAttribute $s1AdminService configRepository]
set props [$AdminConfig showAttribute $configRepository properties]
set foundAllowConfigOverwrites ""
if {$props != "{}"} {
  foreach prop $props {
    if {[AdminConfig showAttribute $prop name] == "allowConfigOverwrites"} {
      set foundAllowConfigOverwrites $prop
      break
    }
  }
}

if {$foundAllowConfigOverwrites == ""} {
  $AdminConfig create Property $configRepository {{name allowConfigOverwrites} {value true}}
} else {
  $AdminConfig modify $foundAllowConfigOverwrites {{value true}}
}

```

```
$AdminConfig save
```

Using Jython:

```

s1AdminService = AdminConfig.getid('/Server:dmgr/AdminService/')
configRepository = AdminConfig.showAttribute(s1AdminService, 'configRepository')
props = AdminConfig.showAttribute(configRepository, 'properties')
foundAllowConfigOverwrites = ''
if props != '[]':
    properties = props[1:len(props)-1].split(' ')
    for prop in properties:
        name = AdminConfig.showAttribute(prop, 'name')
        if name == 'allowConfigOverwrites':
            foundAllowConfigOverwrites = prop
            break

if len(foundAllowConfigOverwrites) != 0:

```

```
AdminConfig.modify(foundAllowConfigOverwrites, [['value', 'true']])
else:
AdminConfig.create('Property', configRepository, [['name', 'allowConfigOverwrites'], ['value', 'true']])
```

```
AdminConfig.save()
```

2. Restart the deployment manager. From the bin directory of the deployment manager profile, run the following:

On Windows platforms:

```
stopManager
startManager
```

On Unix platforms:

```
./stopManager.sh
./startManager.sh
```

3. Allow configuration overwrite, for example:

Using Jacl:

```
$AdminConfig setSaveMode overwriteOnConflict
```

Using Jython:

```
AdminConfig.setSaveMode('overwriteOnConflict')
```

Part 3. Coexisting

Chapter 8. Coexisting

This topic is a starting point for finding information about which coexistence scenarios are supported, and how to set up the scenarios.

- Read about “Coexistence support.”

This topic discusses which coexistence scenarios are supported.

- “Setting up Version 4.0.x and Version 6 coexistence” on page 103.

This task describes how to install a Version 6 product to coexist with another installation instance of Version 4.x.

- “Setting up Version 5 and Version 6 coexistence” on page 104.

This task describes how to install a Version 6 product to coexist with another installation instance of Version 5.x.

- “Setting up Version 6 coexistence” on page 105.

This task describes how to install a Version 6 product to coexist with another installation instance of Version 6.

Coexistence support

Coexistence, as it applies to WebSphere Application Server products, is the ability of multiple installations of WebSphere Application Server to run on the same machine at the same time. Multiple installations include multiple versions and multiple instances of one version. Coexistence also implies various combinations of Web server interaction.

Version 6.0 WebSphere Application Server products can coexist with the following supported versions:

- IBM WebSphere Application Server Advanced Server Single Edition and Advanced Edition, Version 4.0.2 and later
- IBM WebSphere Business Integration Server Foundation Edition, Version 4.1 and later
- IBM WebSphere Application Server, Version 5.0.0 and later
- IBM WebSphere Application Server Network Deployment, Version 5.0.0 and later
- IBM WebSphere Application Server Enterprise, Version 5.0.0 and later
- IBM WebSphere Application Server, Version 5.1.0 and later
- IBM WebSphere Application Server Network Deployment, Version 5.1.0 and later

All combinations of V4.x products, V5.x products, and V6.0 products can coexist so long as there are no port conflicts.

V4.0.2 and later products can coexist with the Version 6.0 WebSphere Application Server clients:

Table 2. Multiversion WebSphere Application Server clients coexistence scenarios

Installed product	WebSphere Application Server clients, V6
IBM WebSphere Application Server Advanced Single Server Edition, Version 4.0.2 and later	Supported
IBM WebSphere Application Server Advanced Edition, Version 4.0.2 and later	Supported
IBM WebSphere Business Integration Server Foundation Edition, Version 4.1 and later	Supported
IBM WebSphere Application Server, Version 5.0.0	Supported
IBM WebSphere Application Server Network Deployment, Version 5.0.0	Supported
IBM WebSphere Business Integration Server Foundation, Version 5.0.0	Supported

Table 2. Multiversion WebSphere Application Server clients coexistence scenarios (continued)

Installed product	WebSphere Application Server clients, V6
IBM WebSphere Application Server, Version 5.0.1	Supported
IBM WebSphere Application Server Network Deployment, Version 5.0.1	Supported
IBM WebSphere Business Integration Server Foundation, Version 5.0.1	Supported
IBM WebSphere Application Server, Version 5.0.2	Supported
IBM WebSphere Application Server Network Deployment, Version 5.0.2	Supported
IBM WebSphere Business Integration Server Foundation, Version 5.0.2	Supported
IBM WebSphere Application Server, Version 5.1	Supported
IBM WebSphere Application Server Network Deployment, Version 5.1	Supported

V4.0.2 and later products and V5.x products can coexist with V6 products:

Table 3. Multiversion coexistence scenarios

Installed product	V5.x			V6		
	Application Server	Network Deployment	Enterprise / WBISF	Application Server	Network Deployment	Express
Advanced Single Server Edition V4.0.2 and later	Supported	Supported	Supported	Supported	Supported	Supported
Advanced Edition V4.0.2 and later	Supported	Supported	Supported	Supported	Supported	Supported
Enterprise Edition V4.1 and later	Supported	Supported	Supported	Supported	Supported	Supported
WebSphere Application Server clients V4.0.x	Supported	Supported	Supported	Supported	Supported	Supported
WebSphere Application Server, V5.0.0	Supported	Supported	Supported	Supported	Supported	Supported
Network Deployment V5.0.0	Supported	Supported	Supported	Supported	Supported	Supported
Enterprise, V5.0.0	Supported	Supported	Supported	Supported	Supported	Supported
WebSphere Application Server clients V5.0.0	Supported	Supported	Supported	Supported	Supported	Supported
WebSphere Application Server V5.0.1	Supported	Supported	Supported	Supported	Supported	Supported
Network Deployment V5.0.1	Supported	Supported	Supported	Supported	Supported	Supported
Enterprise V5.0.1	Supported	Supported	Supported	Supported	Supported	Supported

Table 3. Multiversion coexistence scenarios (continued)

Installed product	V5.x			V6		
	Application Server	Network Deployment	Enterprise / WBISF	Application Server	Network Deployment	Express
WebSphere Application Server clients V5.0.1	Supported	Supported	Supported	Supported	Supported	Supported
WebSphere Application Server V5.0.2	Supported	Supported	Supported	Supported	Supported	Supported
Network Deployment V5.0.2	Supported	Supported	Supported	Supported	Supported	Supported
Enterprise V5.0.2	Supported	Supported	Supported	Supported	Supported	Supported
WebSphere Application Server clients V5.0.2	Supported	Supported	Supported	Supported	Supported	Supported
WebSphere Application Server V5.1	Supported	Supported	Supported	Supported	Supported	Supported
Network Deployment V5.1	Supported	Supported	Supported	Supported	Supported	Supported
Integration Server V5.1	Supported	Supported	Supported	Supported	Supported	Supported
WebSphere Application Server clients V5.1	Supported	Supported	Supported	Supported	Supported	Supported

In addition to multiversion coexistence, WebSphere Application Server also lets you install multiple times on one machine (multiple installation instances), or install once and have multiple profiles.

Multiple Version 6 instances on one machine include:

- Multiple Application Server instances from multiple installations of the WebSphere Application Server product
- Multiple Application Server profiles from a single installation of the WebSphere Application Server product

Setting up Version 4.0.x and Version 6 coexistence

You must migrate prerequisite and corequisite programs to the levels required by WebSphere Application Server, Version 6. You must also identify ports in use in Version 4.0.x before you begin the Version 6 installation, to avoid possible conflicts during coexistence. The first two steps in this task describe these activities.

You can install WebSphere Application Server Version 4.0.x and Version 6 on the same node. Change the ports to avoid conflicts during profile creation.

Silent installation also supports configuring for coexistence silently. You can specify non-conflicting port assignments in the options response file.

By default, there are port conflicts between Version 4.0.x and Version 6 that you must resolve. Also, if you migrate more than two Version 4.0.x nodes to Version 6, there are port conflicts that you must resolve, as described in the “Setting up Version 6 coexistence” on page 105 topic.

1. Migrate prerequisite and corequisite programs to the levels required by WebSphere Application Server, Version 6.

Refer to the IBM WebSphere Application Server supported hardware and software site for current requirements.

2. Resolve port conflicts.

Refer to the Port number settings in WebSphere Application Server versions topic to see a list of default port numbers, and where they are defined.

Inspect the configuration of the previous version:

- **For WebSphere Application Server Advanced Single Server Edition, Version 4.0.x:** Inspect the `server-cfg.xml` file to get port values for the configuration.
- **For WebSphere Application Server Advanced Edition, Version 4.0.x:** Inspect the `admin.config` file to get port values for the configuration. When the administrative server is running, use this command:

```
xmlConfig -export config.xml -nodeName theNodeName
```

Review the `config.xml` file to look for `<node action="update" name="nodename">` to find the appropriate node and port number assignments in the file. The Version 6 installation wizard will display the default set of coexistence port numbers during profile creation time. Change the values to ports that are not in use. The installation wizard uses whatever values you approve.

3. Associate a Web server with each WebSphere Application Server.
 - Use a separate Web server for each WebSphere Application Server.
 - a. Create a Web server instance using the Web server documentation.
 - b. Use the standalone Web server plug-in installation wizard to install the appropriate Version 6 Web server plug-in. This wizard is provided on a separate CD from the WebSphere Application Server Installation CD. The Web server plug-in configuration file, `plugin_cfg.xml`, is now configurable using the administrative console.
 - Use the same Web server for both WebSphere Application Server versions.

To use the same Web server for both Application Server versions, you must first upgrade the Web server to the common level supported by both versions of the application server.

Follow this procedure to use the same Web server for both WebSphere Application Server versions.

 - a. Install the appropriate Version 6 Web server plug-in using the standalone Web server plug-in installation wizard that is provided on a separate CD from the WebSphere Application Server Installation CD.
 - b. Replace the original `plugin-cfg.xml` file of the Version 6 installation with the master file.
4. Fix any problems with environmental variables on Windows platforms. For example, installing WebSphere Application Server, Version 6 updates the system variable `PATH`, potentially affecting tools with the same name across installations. To run tools with conflicting names, alter the `PATH` environment variable in a command window and place the directory for the former installation before the directory for the latter installation. For example, `PATH=E:\WebSphere\AppServer\40\bin;%PATH%`. Then, invoke the tools from the `bin` directory.

Setting up Version 5 and Version 6 coexistence

This task describes how to install a Version 6 product to coexist with another installation instance of Version 5.x.

After installing the WebSphere Application Server product, you can install it again on the same machine.

Each installation of the base product is a stand-alone Application Server (`server1`) with its own set of unique configuration files.

Be aware of multiple instance limitations:

- If you install more than one instance, the most recent installation is the only one in the operating system registry.
- If you install more than two instances, the third and subsequent installations require that you change all port numbers on the coexistence panel, to avoid potential conflicts.
- To uninstall a registered product instance, always use the operating system remove program function, such as the Add/Remove Program utility on Windows platforms. To uninstall an unregistered instance, use the **Uninstall.exe** or **uninstall** command in the *install_root/_uninstall* directory that matches the instance you intend to remove.

Reasons to use multiple installation instances include:

- You can achieve complete isolation between each WebSphere Application Server instance. You can uninstall one instance independently of the others.
- You can install the base WebSphere Application Server more than once on the same machine.
- You can install the base V5.x and V6 WebSphere Application Server on the same machine.

Reasons to not use multiple installation instances include:

- The machine might have a hard disk space constraint.
- You can use the operating system registry to locate the last installed instance of a WebSphere Application Server product only.

When you install any product a second time, the last installation is the one that appears in the registry.

- Uninstalling the last instance removes any record of the product in the registry.

Suppose you have installed three instances of the base WebSphere Application Server product. You use the remove program function of the operating system to uninstall the registered third copy of the base product. A registry record no longer exists that indicates the existence of the other two installation instances. Other applications cannot use a query of the operating system registry to detect the presence of either base WebSphere Application Server product instance.

Use the Profile creation wizard to installing the base WebSphere Application Server product once and creating multiple profiles.

Use the following procedure to install multiple installation instances.

1. Use the installation wizard to install another installation. If you intend to share a single Web server among installations, install the appropriate Version 6 Web server plug-in using the standalone Web server plug-in installation wizard that is provided on a separate CD from the WebSphere Application Server Installation CD.
2. Share a Web server among multiple installation instances.
 - a. Use the Plug-in Installation wizard to select a Web server plug-in.
 - b. Use the administrative console to generate the plug-in configuration files for every installation instance and to then merge them into one master configuration.
 - c. Use the administrative console to replace the original `plugin-cfg.xml` file with the master file on the Web server.

You can access samples from only one of the installation instances.

3. Change port assignments in configuration files if you have a node that you cannot start because of port conflicts.

Setting up Version 6 coexistence

This task describes how to install a V6 product to coexist with concurrent V6.0 profiles.

After installing the WebSphere Application Server product, you can install it again on the same machine.

Select the new installation option from the installation wizard panel, to install a new instance instead of adding features to the last installation, and to install into a separate profile directory.

Each installation of the product installs the core product files.

You can install once and use the Profile creation wizard to create multiple Application Server processes.

Use the following procedure to install multiple copies of the core product files.

1. Use the installation wizard to install another set of the core product files.
2. Create additional server profiles and processes, using the Profile creation wizard.
3. If you have a node that you cannot start because of port conflicts, change port assignments to non-conflicting ports in configuration files by editing the *profiles_install_root/profile_name/config/cells/cell_name/nodes/node_name/serverindex.xml* file.

Chapter 9. Configuring ports

This topic discusses configuring ports, particularly in coexistence scenarios.

1. Review “Port number settings in WebSphere Application Server versions.”

This topic provides reference information about identifying port numbers in versions of WebSphere Application Server, as a means of determining port conflicts that might occur when you intend for an earlier version to coexist with Version 6.

2. You can change port settings on the port assignment panel while using the Installation wizard or the Profile creation wizard.

3. After installation, edit the `profiles_install_root/profile_name/config/cells/cell_name/nodes/node_name/serverindex.xml` file to change the port settings, or use scripting to change the values.

See the *Administration: Operations* PDF for more information.

Port number settings in WebSphere Application Server versions

This topic provides reference information about identifying port numbers in versions of WebSphere Application Server, as a means of determining port conflicts that might occur when you intend for an earlier version to coexist or interoperate with Version 6.

Version 6 port numbers

Table 4. Port definitions for WebSphere Application Server Version 6

Port name	WebSphere Application Server	File
	Value	
HTTP_TRANSPORT	9080	
HTTP Admin Console Port (HTTP_TRANSPORT_ADMIN)	9060	serverindex.xml and virtualhosts.xml
HTTPS Transport Port (HTTPS_TRANSPORT)	9443	
HTTPS Admin Console Secure Port (HTTPS_TRANSPORT_ADMIN)	9043	
BOOTSTRAP_ADDRESS	2809	
SOAP_CONNECTOR_ADDRESS	8880	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9401	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9403	
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9402	
ORB_LISTENER_ADDRESS	9100	serverindex.xml
DCS_UNICAST_ADDRESS	9353	
SIB_ENDPOINT_ADDRESS	7276	
SIB_ENDPOINT_SECURE_ADDRESS	7286	
SIB_MQ_ENDPOINT_ADDRESS	5558	
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5578	
Internal JMS Server (JMSSERVER_SECURITY_PORT)	5557	
DRS_CLIENT_ADDRESS	7873	

Table 4. Port definitions for WebSphere Application Server Version 6 (continued)

Port name	WebSphere Application Server	File
	Value	
IBM HTTP Server Port	80	virtualhosts.xml, plugin-cfg.xml, and <i>IHSinstall_root/conf/httpd.conf</i>
IBM HTTP Server Admin Port	8008	<i>IHSinstall_root/conf/admin.conf</i>
NODE_MULTICAST_IPV6_DISCOVERY_ADDRESS	5001	serverindex.xml

Version 5.x port numbers

Version 4.0.x port numbers

For WebSphere Application Server Advanced Single Server Edition, Version 4.0.x: Inspect the `server-cfg.xml` file to find the Web container HTTP transports port values for the configuration.

For WebSphere Application Server Advanced Edition, Version 4.0.x: When the administrative server is running, use this command to extract the configuration from the database:

```
xmlConfig -export config.xml -nodeName theNodeName
```

Look for the Web container HTTP transports port assignments.

Table 5. Port definitions for WebSphere Application Server V4.0.x

Port name	Value	Advanced Edition	IBM WebSphere Business Integration Server Foundation Edition	Advanced Single Server Edition
		File		
bootstrapPort	900			
lsdPort	9000	admin.config	admin.config	
LSDSSLPort	9001			
HTTP transport port	9080			
HTTPS transport port	9443			server-cfg.xml
Admin Console HTTP transport port	9090	database	database	
ObjectLevelTrace	2102			
diagThreadPort	7000			

Part 4. Interoperating

Chapter 10. Interoperating

WebSphere Application Server Version 6 is generally interoperable with WebSphere Application Server Versions 4.0.x and 5.x. However, there are specific requirements to address for each version. In general, IBM recommends that you apply the latest fix level to support interoperability. If this is not possible, then the following interim fixes can be used to support your environment.

1. Apply required interim fixes.

Table 6. Interim fixes to apply to Version 4.0.x

Interim fix	Version 4.0.1	Version 4.0.2	Version 4.0.3	Version 4.0.5	Version 4.0.6	Version 4.0.7
PQ60074	Apply	Apply				
PQ60336	Apply	Apply				
PQ63548	Apply	Apply	Apply			
PQ88648 (which requires PQ88653)				Apply	Apply	Apply
PQ88973			Apply	Apply	Apply	

Table 7. Interim fixes to apply to Version 5.0.x

Interim fix	Version 5.0.0	Version 5.0.1	Version 5.0.2
PQ88973	Apply	Apply	Apply
PQ89426 (which requires PQ88653)			Apply (or move to 5.0.2.8)

Table 8. Interim fixes to apply to Version 5.1.x

Interim fix	Version 5.1.0	Version 5.1.1
PQ84384	Apply (or move to 5.1.0.4 or higher)	

All fixes are available on the Support site for WebSphere Application Server products. There is a link to the Support site for WebSphere Application Server products at the bottom of each information center topic. Scroll all the way to the bottom of each page to see the link.

Interim fix PQ60074

An Object Request Broker (ORB) fix that supports V6 naming client access to the Version 4.0.x name server. A down-level client has no problem accessing a V6 name server, even when using corbaloc.

Interim fix PQ60336

An ORB fix to reconcile `java.math.BigDecimal` and other class differences in IBM Software Development Kits 130 and 131.

Note: This fix does not apply to IBM Software Development Kits on Solaris platforms.

Interim fix PQ63548

A fix to correct problems that might occur when passing embedded valueTypes between WebSphere Application Server releases.

The best solution is to upgrade all your installations to the latest release and PTF levels, such as Version 4.0.4, which do not require this fix. If this solution is not possible, apply the fix to your version.

Symptoms include `org.omg.CORBA.MARSHAL` exceptions when passing embedded valueTypes across the versions. Sometimes, other symptoms might mask `org.omg.CORBA.MARSHAL` exceptions, which makes them difficult to identify.

If symptoms reoccur in spite of the fix, re-export existing IORs.

Interim fixes PQ88648 (V4), PQ89426 (V5.0.2), PQ84384(V5.1.0):

The transaction service is changed so that when a transaction is marked for rollbackOnly in a subordinate server, the superior server will be informed. This will allow applications running in the superior server to detect this status change.

Interim fix PQ88973

An interim fix to upgrade the Software Development Kit (SDK) used by the Version 5.0.x client. The evolution of a number of core classes causes interoperability errors between a WebSphere Application Server, Version 5.0.x client and a Version 6 server.

You might see the following message when running an interoperability scenario between a WebSphere Application Server, Version 5.0.x client and a WebSphere Application Server, Version 6 server:

```
java.rmi.MarshalException:
  CORBA MARSHAL 0x4942f89a No;
  nested exception is: org.omg.CORBA.MARSHAL:
  Unable to read value from underlying bridge :
  Invalid start_value valuetag: c
minor code: 4942F89A
completed: No
```

A number of core classes evolved between Software Development Kit (SDK) 1.3.x and SDK 1.4.x. You can experience problems interoperating with WebSphere Application Server, Version 6, which uses SDK 1.4.x.

The recommended response is to upgrade SDK 1.3.1 to a newer Service Release (SR). The SDK Service Release updates are available at the following IBM support sites:

- V5.0.2 Cumulative Fix for SDKs
- 1.3.1 Java SDK, Java Tech Edition for WebSphere Application Server V4

2. Follow the required guidelines for V4.0.x and V5.0.2.

Table 9. Guidelines to apply for Version 5.x and Version 4.0.x

Guideline	Version 5.x	Version 4.0.x
1	Apply (V5.0.2 only)	Apply
2		Apply
3		Apply
4		Apply
5		Apply
6	Apply	Apply

Guideline 1:

Set the following JVM properties:

```
com.ibm.ejs.jts.jts.ControlSet.nativeOnly=false
com.ibm.ejs.jts.jts.ControlSet.interoperabilityOnly=true
```

Always apply this guideline to V4.0.x. For V5.0.2, apply this guideline in addition to applying interim fixes (or moving to V5.0.2.8).

Guideline 2:

Make required naming changes to support Version 4.0.x client access to Version 6 enterprise beans. There are several ways to make it work, such as:

- Updating the namebindings.xml file if you do not use the Version 6 migration tools, but have installed Version 4.0.x applications on Version 6. To allow Version 4.0.x client access to the applications, add additional information to the bind information in the Version 6 namespace to make the old JNDI names work. Add the information to the namebindings.xml configuration file at the cell level using the administrative console.

Note: Applications that you migrate to Version 6 using the WASPreUpgrade and WASPostUpgrade migration tools already have this update.

After federating an application server node into a deployment manager cell, you cannot use the migration tools. To use these tools again, remove the node from the cell, use the tools, and add the node to the cell again.

- Calling Version 6 enterprise beans directly using the naming path that includes the server on which the enterprise beans are running, such as `cell/node/server1/some/ejb/name`, for example.
- Using the Version 4.0.x client `java:/comp` location to find Version 6 enterprise beans.

Guideline 3:

Be aware of administrative console limitations.

You cannot use the administrative interfaces for Version 6 to administer a Version 4.0.x administrative server. Likewise, you cannot use a Version 4.0.x administrative console to administer a Version 6 environment. If you use the administrative console on a remote machine to administer Version 4.0.x WebSphere Application Server domains, migrating any of the nodes or domains to Version 6 renders the remote administration console ineffective for administering any Version 6 environment.

Guideline 4:

Use Secure Sockets Layer Version 3 (SSL v3) for secure connections when interoperating . You cannot use Common Secure Interoperability Version 2 (CSIv2) for interoperability, because Version 4.0.x does not support CSIv2.

Guideline 5:

(This guideline applies only to V4.0.1.) Be aware of limitations when calling WorkLoad Management (WLM)-enabled enterprise beans.

Some clients cannot call WLM-enabled enterprise beans on remote clusters when there is a local WLM-enabled enterprise bean of the same name. If there is a cluster local to the client with the same enterprise bean as the remote cluster that the client is trying to call, the client ends up talking to the local cluster.

Guideline 6:

Be aware of the level of WebSphere Application Server in which each function you use is supported. Applications that you intend to be interoperable must only use function that is supported by all levels of WebSphere Application Server in the cluster. For example, applications that use the `commonj.timer.TimerManager` resource, which is new for V6, should not be deployed to a cluster including both V5.1 and V6 servers.

This information is dynamic and might be augmented by information in technical articles that are available on the IBM DeveloperWorks WebSphere site. Check the site for the latest information.

Chapter 11. Container interoperability

Container interoperability describes the ability of WebSphere Application Server clients and servers at different versions to successfully negotiate differences in native Enterprise JavaBeans (EJB) finder methods support and Java 2 Platform, Enterprise Edition (J2EE) compliance.

The product uses interoperable versions of some class types to enable interoperability. However, older 4.0.x client and application server versions do not support the interoperability classes, which makes them uninteroperable with versions that use the classes. The system property `com.ibm.websphere.container.portable` remedies this situation by enabling newer versions of the application server to turn off the interoperability classes. This lets a more recent application server return class types that are interoperable with an older client.

Depending on the value of `com.ibm.websphere.container.portable`, application servers at versions 5 and later, and 4.0.3 and later, return different classes for the following:

- Enumerations and collections returned by EJB 1.1 finder methods
- EJBMetaData
- Handles to:
 - Entity beans
 - Session beans
 - Home interfaces

If the property is set to `false`, application servers return the old class types, to enable interoperability with 4.0.2 and earlier. If the property is set to `true`, application servers return the new classes.

The following tables show interoperability characteristics for various version combinations of application servers and clients as well as default property values for each combination.

Interoperability of Version 4.0.x client with Version 5 (and later) application server

Ideally, all 4.0.x clients that use Version 5 or later application servers should be at Version 4.0.3 or later.

Version 5 and later application servers return the interoperability class types by default (`true`). This can cause interoperability problems for distributed clients at versions 4.0.1 or 4.0.2. In particular, problems can occur with collections and enumerations returned by Enterprise JavaBeans Version 1.1 finder methods.

Although it is strongly discouraged, you can set `com.ibm.websphere.container.portable` to `false` on a Version 5 and later application server. This causes the application server to return the old class types, providing interoperability with clients at Version 4.0.2 and earlier. This is discouraged because:

- The Version 5 application server instance would become non-J2EE 1.3 (and later) compliant with regard to handles, home interface handles, and EJBMetaData.
- EJB 1.x finder methods return collection and enumeration objects that do not originate from `ejbportable.jar`.
- Interoperability restrictions still exist with the property set to `false`.
- Version 5 and later client handles to entity beans and home interfaces do not work across domains for the server you set to `false`.

If you would like to use updated Handle classes in EJB 2.x-compliant beans but have one of the older clients (versions 4.0.2 and earlier) installed, set the system property `com.ibm.websphere.container.portable.finder` to `false`. With this setting in place, the Version 5 and later application server uses the updated handles but returns the enumerations and collections that were used in the earlier clients.

Interoperability of client at Version 4.0.2 and earlier with Version 5 (and later) application server

Client at Version 4.0.2 and earlier, using this function	Application server at Version 5 and later, property true (default)	Application server at Version 5 and later, property false
EJBMetaData	Does not work	Works for 4.0.2 client
Handle to session bean	Does not work	Works
Handle to entity bean	Does not work	Does not work across cells
Enumeration returned by EJB 1.x finder method	Does not work	Works
Collection returned by EJB 1.x finder method	Does not work	Works
Handle to home interface	Does not work	Does not work across cells

If you would like to use updated Handle classes in EJB 2.x-compliant beans but have one of the older clients (versions 4.0.2 and earlier) installed, set the system property `com.ibm.websphere.container.portable.finder` to `false`. With this setting in place, the Version 5 and later server uses the new Handle classes but returns the older enumeration and collection classes.

Interoperability of client at Version 4.0.3 and later with Version 5 and later application server

Clients at Version 4.0.3 and later work well with Version 5 and later application servers. However, if you set the `com.ibm.websphere.container.portable` to `false`, client handles to entity beans and home interfaces do not work across domains for the server you set to `false`.

Client at Version 4.0.3 and later, using this function	Application server at Version 5 and later, property true (default)	Application server at Version 5 and later, property false
EJBMetaData	Works	Works
Handle to session bean	Works	Works
Handle to entity bean	Works	Does not work across cells
Enumeration returned by EJB 1.x finder method	Works	Works
Collection returned by EJB 1.x finder method	Works	Works
Handle to home interface	Works	Does not work across cells

Interoperability of Version 5 and later client with Version 4.0.x application server

Clients at Version 5 and later work well with Version 4.0.3 application servers if you set `com.ibm.websphere.container.portable` to `true`. Client handles to entity beans and home interfaces do not work across domains for any Version 4.0.3 server with `com.ibm.websphere.container.portable` at the default value, `false`. Version 5 client handles to application servers at Version 4.0.2 and earlier also have restrictions.

Client at Version 5 and later, using this function	Application server at Version 4.0.3, property true	Application server at Version 4.0.3, property false (default)	Application server at Version 4.0.2 or earlier
EJBMetaData	Works	Works	Works for 4.0.2 server only
Handle to session bean	Works	Works	Works
Handle to entity bean	Works	Does not work across domains	Does not work across domains

Client at Version 5 and later, using this function	Application server at Version 4.0.3, property true	Application server at Version 4.0.3, property false (default)	Application server at Version 4.0.2 or earlier
Enumeration returned by EJB 1.x finder method	Works	Works	Works
Collection returned by EJB 1.x finder method	Works	Works	Works
Handle to home interface	Works	Does not work across domains	Does not work across domains

Interoperability of zSeries Version 4.0.x client with Version 5 and later application server

The only valid configuration for container interoperability with zSeries Version 4.0.x clients is the default configuration for the Version 5 application server.

Interoperability of Version 5 and later client with zSeries Version 4.0.x application server

Version 5 clients should work with a zSeries Version 4.0.x application server with the correct interoperability fixes described in the zSeries documentation. The interoperability characteristics should be the same as for a Version 4.0.3 distributed application server with the property set to true.

Client at Version 5 and later, using this function	zSeries application server at Version 4.0.x
EJBMetaData	Works
Handle to session bean	Works
Handle to entity bean	Works
Enumeration returned by EJB 1.x finder method	Works
Collection returned by EJB 1.x finder method	Works
Handle to home interface	Works

Chapter 12. Web Services-Interoperability Basic Profile

The *Web Services-Interoperability (WS-I) Basic Profile* is a set of non-proprietary Web services specifications that promote interoperability.

WebSphere Application Server conforms to the WS-I Basic Profile 1.1.

The WS-I Basic Profile is governed by a consortium of industry-leading corporations, including IBM, under direction of the WS-I Organization. The profile consists of a set of principles that relate to bringing about open standards for Web services technology. All organizations that are interested in promoting interoperability among Web services are encouraged to become members of the Web Services Interoperability Organization.

Several technology components are used in the composition and implementation of Web services, including messaging, description, discovery, and security. Each of these components are supported by specifications and standards, including SOAP 1.1, Extensible Markup Language (XML) 1.0, HTTP 1.1, Web Services Description Language (WSDL) 1.1, and Universal Description, Discovery and Integration (UDDI). The WS-I Basic Profile specifies how these technology components are used together to achieve interoperability, and mandates specific use of each of the technologies when appropriate. You can read more about the WS-I Basic Profile at the WS-I Organization Web site.

Each of the technology components have requirements that you can read about in more detail at the WS-I Organization Web site. For example, support for Universal Transformation Format (UTF)-16 encoding is required by WS-I Basic Profile. UTF-16 is a kind of Unicode encoding scheme using 16-bit values to store Universal Character Set (UCS) characters. UTF-8 is the most common encoding that is used on the Internet; UTF-16 encoding is typically used for Java and Windows product applications; and UTF-32 is used by various Linux and Unix systems. Unlike UTF-8, UTF-16 has issues with big-endian and little-endian, and often involves Byte Order Mark (BOM) to indicate the endian. BOM is mandatory for UTF-16 encoding and it can be used in UTF-8.

The following table summarizes some of the properties of each UTF:

Bytes	Encoding form
EF BB BF	UTF-8
FF FE	UTF-16, little-endian
FE FF	UTF-16, big-endian
00 00 FE FF	UTF-32, big-endian
FF FE 00 00	UTF-32, little-endian

BOM is written prior to the XML text, and it indicates to the parser how the XML is encoded. The XML declaration contains the encoding, for example: `<?xml version=xxx encoding="utf-xxx"?>`. BOM is used with the encoding to determine how to interpret the XML. Here is an example of a SOAP message and how BOM and UTF encoding are used:

```
POST http://www.whitemesa.net/soap12/add-test-rpc HTTP/1.1
Content-Type: application/soap+xml; charset=utf-16; action=""
SOAPAction:
Host: localhost: 8080
Content-Length: 562
```

```
0xFF0xFE<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2002/12/soap-envelope"
  xmlns:soapenc="http://www.w3.org/2002/12/soap-encoding"
  xmlns:tns="http://whitemesa.net/wsd1/soap12-test"
  xmlns:types="http://whitemesa.net/wsd1/soap12-test/encodedTypes"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soap:Body>
  <q1:echoString xmlns:q1="http://soapinterop.org/">
    <inputString soap:encodingStyle="http://example.org/unknownEncoding"
      xsi:type="xsd:string">
      Hello SOAP 1.2
    </inputString>
  </q1:echoString>
</soap:Body>
</soap:Envelope>
```

In the example code, 0xFF0xFE represents the byte codes, while the `<?xml/>` declaration is the textual representation.

To learn more about the WS-Basic profile, including scenarios, UTF and BOM, see the *Developing and Deploying Applications* PDF.

Chapter 13. Interoperability issues for security

To have interoperability of Security Authentication Service (SAS) between C++ and WebSphere Application Server, use the Common Secure Interoperability Version 2 (CSIv2) authentication protocol over Remote Method Invocation over the Internet Inter-ORB Protocol (RMI-IIOP).

Chapter 14. Interoperating with a C++ common object request broker architecture client

You can achieve interoperability of Security Authentication Service between the C++ Common Object Request Broker Architecture (CORBA) client and WebSphere Application Server using Common Secure Interoperability Version 2 (CSIv2) authentication protocol over Remote Method Invocation over the Internet Inter-ORB Protocol (RMI-IIOP). The CSIv2 security service protocol has authentication, attribute and transport layers. Among the three layers, transport authentication is conceptually simple, however, cryptographically based transport authentication is the strongest. WebSphere Application Server Enterprise has implemented the transport authentication layer, so that C++ secure CORBA clients can use it effectively in making CORBA clients and protected enterprise bean resources work together.

Security authentication from non-Java based C++ client to enterprise beans. WebSphere Application Server supports security in the CORBA C++ client to access protected enterprise beans. If configured, C++ CORBA clients can access protected enterprise bean methods using client certificate to achieve mutual authentication on WebSphere Application Server Enterprise applications.

To support the C++ CORBA client in accessing protected enterprise beans:

- Create an environment file for the client, such as `current.env`. Set the variables listed below (`security_sslKeyring`, `client_protocol_user`, `client_protocol_password`) in the file.
- Point to the environment file using the fully qualified path name through the environment variable `WAS_CONFIG_FILE`. For example, in the test shell script `test.sh`, export `WAS_CONFIG_FILE=/WebSphere/V5R0M0/AppServer/bin/current.env`.

C++ security setting	Description
<code>client_protocol_password</code>	Specifies the password for the user ID.
<code>client_protocol_user</code>	Specifies the user ID to be authenticated at the target server.
<code>security_sslKeyring</code>	Specifies the name of the RACF keyring the client will use. The keyring must be defined under the user ID that is issuing the command to run the client.

To support the C++ CORBA client in accessing protected enterprise beans:

1. Obtain a valid certificate to represent the client and export its public key to the target enterprise bean server.

A valid certificate is needed to represent the C++ client. Request a certificate from the certificate authority (CA) or create a self-signed certificate for testing purposes.

Use the Key Management Utility from the Global Security Kit (GSKit) to extract the public key from the personal certificate and save it in the `.arm` format. For details, see the *Security Guide* PDF.

2. Prepare a truststore file for WebSphere Application Server.

Add the extracted client public key in the `.arm` file from the client to the server key truststore file. The server can now authenticate the client.

Note: This is done by invoking the Key Management Utility through `keyman.bat` or `keyman.sh` from WebSphere Application Server installation.

For details, see the *Security Guide* PDF.

3. Configure WebSphere Application Server to support SSL as the authentication mechanism.
 - a. Start the administrative console.
 - b. Locate the application server that has the target enterprise bean deployed and configure it to use SSL client certificate authentication.

If it is a base installation, complete the following steps:

- 1) Click **Security > Global security**. Under Authentication, click **Authentication protocol > CSiv2 inbound authentication**. Select **Supported** for the Basic authentication and Client certificate authentication options. Leave the rest of the options as defaults.
- 2) Click **Security > Global security**. Under Authentication, click **Authentication protocol > CSiv2 inbound transport** and verify that the **SSL-supported** option is selected.

If it is a Network Deployment setting, complete the following steps:

- 1) Click **Server > Application Server > *server_name_where_the_EJB_resides***. Under security, click **Server security**. Under Additional properties, click **CSI inbound authentication**. Select **Supported** for the Basic authentication and Client certificate authentication options. Leave the rest of the options as defaults.
- 2) Click **Server > Application Server > *server_name_where_the_EJB_resides***. Under security, click **Server security**. Under Additional properties, click **CSI inbound transport**. Verify that the **SSL-Supported** option is selected.

For details, see the *Security Guide* PDF.

- c. Restart the application server.

The WebSphere Application Server is ready to take a C++ CORBA security client and a mutually authenticated server and client by using SSL in the transport layer.

4. Configure the C++ CORBA client to use a certificate in performing the mutual authentication.

Client users are accustomed to using property files in their applications because they are helpful in specifying configuration settings. The following list presents important C++ security settings:

C++ security setting	Description
com.ibm.CORBA.bootstrapHostName=ricebella.austin.ibm.com	Specifies the target host name.
com.ibm.CORBA.securityEnabled=yes	Enables security.
com.ibm.CSI.performTLClientAuthenticationSupported=yes	Ensures client is supporting mutual authentication by certificate
com.ibm.CSI.performTransportAssocSSLTLSSupported=yes	Ensures SSL is used, not TCP/IP
com.ibm.ssl.keyFile=C:/ricebella/etc/DummyKeyRingFile.KDB	Specifies which key database file to use.
com.ibm.ssl.keyPassword=WebAS	Specifies the password for opening the key database file. WebSphere Application Server supports a utility called PasswordEncode4cpp to encode the plain password.
com.ibm.CORBA.translationEnabled=1	Enables the valueType conversion.

To use the property files in running a C++ client, an environment variable WASPROPS, is used to indicate where a property file or a list of property files exist.

For the complete set of C++ client properties, see the sample property file `sccClient.props`, which is shipped with the product located in the `install_root\profiles\profile_name\etc` directory.

Chapter 15. Interoperating with previous product versions

IBM WebSphere Application Server, Version 5.x or later interoperates with the previous product versions (such as Version 4 and Version 3.5). Interoperability is achieved only when the Lightweight Third Party Authentication (LTPA) authentication mechanism and Lightweight Directory Access Protocol (LDAP) user registry are used. Credentials derived from Simple WebSphere Authentication Mechanisms (SWAM) are not forwardable.

1. Enable security with the LTPA authentication mechanism and the LDAP user registry. Make sure that the same LDAP user registry is shared by all the product versions.
2. Extract and add Version 5 server certificates into the server key ring file of the previous version.
 - a. Open the Version 5 server key ring file using the key management utility (iKeyman) and extract the server certificate to a file.
 - b. Open the server key ring of the previous product version, using the key management utility and add the certificate extracted from product Version 5.
3. Extract and add Version 5 server certificates into the server key ring file of the previous version.
 - a. Open the Version 5 server key ring file using the key management utility (iKeyman) and extract the server certificate to a file.
 - b. Open the server key ring of the previous product version, using the key management utility and add the certificate extracted from product Version 5.
4. Extract and add Version 5 trust certificates into the trust key ring file of the previous product version.
 - a. Open the Version 5 trust key ring file using the key management utility and extract the trust certificate to a file.
 - b. Open the trust key ring file of the previous product version using the key management utility and add the certificate extracted from Version 5.
5. If single signon (SSO) is enabled, export keys from the Version 5 product and import them into the previous product version. The Version 4 product requires the fix, PQ61779, and the Version 3.5 product requires the fix, PQ59667, for SSO to function.
6. Verify that the application uses the correct JNDI name. In Version 5, the enterprise beans are registered with long JNDI names like, (top)/nodes/node_name/servers/server_name/HelloHome. Whereas in previous releases, enterprise beans are registered under a root like, (top)/HelloHome. Therefore, EJB applications from previous versions perform a lookup on the Version 5 enterprise beans.

You can also create EJB name bindings in Version 5 that are compatible with the previous version. To create an EJB name binding at the root Version 5, start the administrative console and click **Environment > Naming > Naming Space Bindings > New > EJB > Next**. Complete all the fields and enter a short name (for example, -HelloHome) as the JNDI Name. Click **Next** and **Finish**.
7. Stop and restart all the servers.
8. Make sure that the correct naming bootstrap port is used to perform naming lookup. In previous product versions, the naming bootstrap port is **900**. In Version 5, the bootstrap port is **2809**.

Chapter 16. JNDI interoperability considerations

This section explains considerations to take into account when interoperating with WebSphere Application Server V4.0 and with non-WebSphere Application Server JNDI clients. Also, the way resources from MQSeries must be bound to the name space changed after V4.0 and is described below.

Interoperability with WebSphere Application Server V4.0

- **EJB clients running on WebSphere Application Server V4.0 accessing EJB applications running on WebSphere Application Server V5 or V6**

Applications migrated from previous versions of WebSphere Application Server may still have clients still running in a previous release. The default initial JNDI context for EJB clients running on previous versions of WebSphere Application Server is the cell persistent root (legacy root). The home for an enterprise bean deployed in version 5 or 6 is bound to its server's server root context. In order for the EJB lookup name for down-level clients to remain unchanged, configure a binding for the EJB home under the cell persistent root.

- **EJB clients running on WebSphere Application Server V5 or V6 accessing EJB applications running on WebSphere Application Server V4.0 servers**

The default initial context for a WebSphere Application Server V4.0 server is the correct initial context. Simply look up the JNDI name under which the EJB home is bound.

Note: To enable WebSphere Application Server V5 or V6 clients to access version 4.x servers, the down-level installations must have e-fix PQ60074 installed.

EJB clients running in an environment other than WebSphere Application Server accessing EJB applications running on WebSphere Application Server V5 or V6 servers

When an EJB application running in WebSphere Application Server V5 or V6 is accessed by a non-WebSphere Application Server EJB client, the JNDI initial context factory is presumed to be a non-WebSphere Application Server implementation. In this case, the default initial context will be the cell root. If the JNDI service provider being used supports CORBA object URLs, the corbaname format can be used to look up the EJB home. The construction of the stringified name depends on whether the object is installed on a single server or cluster, as shown below.

- **Single server**

```
initialContext.lookup(  
    "corbaname:iiop:myHost:2809#cell/nodes/node1/servers/server1/myEJB");
```

According to the URL above, the bootstrap host and port are **myHost** and **2809**, and the enterprise bean is installed in a server **server1** in node **node1** and bound in that server under the name **myEJB**.

- **Server cluster**

```
initialContext.lookup(  
    "corbaname:iiop:myHost:2809#cell/clusters/myCluster/myEJB");
```

According to the URL above, the bootstrap host and port are **myHost** and **2809**, and the enterprise bean is installed in a server cluster named **myCluster** and bound in that cluster under the name **myEJB**.

The above lookup will work with any name server bootstrap host and port configured in the same cell.

The above lookup will also work if the bootstrap host and port belongs to a member of the cluster itself. To avoid a single point of failure, the bootstrap server host and port for each cluster member could be listed in the URL as follows:

```
initialContext.lookup(  
    "corbaname:iiop:host1:9810,host2:9810#cell/clusters/myCluster/myEJB");
```

The name prefix **cell/clusters/myCluster/** is not necessary if bootstrapping to the cluster itself, but it will work. The prefix is needed, however, when looking up enterprise beans in other clusters. Name

bindings under the **clusters** context are implemented on the name server to resolve to the server root of a running cluster member during a lookup; thus avoiding a single point of failure.

- **Without CORBA object URL support**

If the JNDI initial context factory being used does not support CORBA object URLs, the initial context can be obtained from the server, and the lookup can be performed on the initial context as follows:

```
Hashtable env = new Hashtable();
env.put(CONTEXT.PROVIDER_URL, "iiop://myHost:2809");
Context ic = new InitialContext(env);
Object o = ic.lookup("cell/clusters/myCluster/myEJB");
```

Binding resources from MQSeries 5.2

In releases previous to WebSphere Application Server V5, the MQSeries jmsadmin tool could be used to bind resources to the name space. When used with a WebSphere Application Server V5 or V6 name space, the resource is bound within a transient partition in the name space and does not persist past the life of the server process. Instead of binding the MQSeries resources with the jmsadmin tool, bind them from the WebSphere Application Server administrative console, under **Resources** in the console navigation tree.

Chapter 17. Application profiling interoperability

The effect of 5.x Compatibility Mode

Application profiling supports *forward* compatibility. Application profiles created in previous versions of WebSphere Application Server (Enterprise Edition 5.0 or WebSphere Business Integration Server Foundation 5.1) can only run in WebSphere Application Server Version 6 if the Application Profiling 5.x Compatibility Mode attribute is turned on. If the 5.x Compatibility Mode attribute is off, Version 5 application profiles might display unexpected behavior. For information about the 5.x Compatibility Mode, see the *Developing and Deploying Applications* PDF.

Similarly, application profiles that you create using WebSphere Application Server Version 6 are not compatible with Version 5 or earlier versions. Even applications configured with application profiles run on Version 6 servers with the Application Profiling 5.x Compatibility Mode attribute turned on cannot interact with applications configured with profiles run on Version 5 servers.

Note: If you select the 5.x Compatibility Mode attribute on the Application Profile Service console page, then tasks configured on J2EE 1.3 applications are not necessarily associated with units of work and can arbitrarily be applied and overridden. This is not a recommended mode of operation and can lead to unexpected deadlocks during database access. Tasks are not communicated on requests between applications that are running under the Application Profiling 5.x Compatibility Mode and applications that are not running under the compatibility mode.

For a Version 6.0 client to interact with applications run under the Application Profiling 5.x Compatibility Mode, you must set the *appprofileCompatibility* system property to **true** in the client process. You can do this by specifying the *-CCDappprofileCompatibility=true* option when invoking the *launchClient* command.

Considerations for a clustered environment

In a clustered environment with mixed WebSphere Application Server product versions and mixed platforms, applications configured with application profiles might exhibit unexpected behavior because previous versions of server members cannot support the application profiling of Version 6.

If a clustered environment contains both Version 5.x and 6.0 server members, and if any applications are configured with application profiles, the Application Profiling 5.x Compatibility Mode attribute must be turned on in Version 6 server members. Still, this cluster can only support Version 5 application profiling behavior. To support applications configured with Version 6 application profiles in a cluster environment, all server members in the cluster must be at the Version 6 level.

WebSphere Application Server Enterprise Edition Version 5.0.2

If you use WebSphere Application Server Enterprise Edition 5.0.2, you must apply WebSphere Application Server Version 5 service pack 7 or later service pack to enable Application Profiling interoperability.

Chapter 18. Interoperating with asynchronous beans

Asynchronous beans support Serialized WorkWithExecutionContext interoperability with objects serialized in 5.0.2 or later.

For more information on migrating to WebSphere Application Server Version 6 from previous product releases, see Migrating product configurations.

1. Install the Version 6 product. Installing the product creates a stand-alone Application Server.
- 2.
3. Start the **First steps** console.
4. Select the **Migration wizard** on the **First steps** console.
5. Use the Migration wizard to migrate the previous release to the Version 6 product.

Chapter 19. Interoperating with schedulers

Schedulers support forward compatibility. Tasks created in previous versions of WebSphere Application Server Enterprise Edition 5.0 or WebSphere Business Integration Server Foundation 5.1 continue to run in WebSphere Application Server, Version 6 schedulers. Tasks that you create using Version 6 are not compatible with product schedulers from Version 5 or earlier. Version 5 schedulers do not run any Version 6 tasks.

All schedulers that are configured to use the same database and tables are considered a clustered scheduler. To guarantee that your tasks will run correctly, all servers in a scheduler cluster must be at the same version. If the servers are at different versions, tasks created with a Version 6 scheduler may not run. If a mixed-Version environment is required for a short period of time, then all scheduler poll daemons should be stopped on all Version 5 servers to allow a Version 6 server to run all tasks. This action allows the Version 6 schedulers to obtain leases and run tasks that have been created with a Version 6 scheduler.

Running tasks created with schedulers prior to Version 5.0.2 is not supported. See the topic, "Interoperating with the Scheduler service," in the WebSphere Application Server Enterprise Edition Version 5.0.2 information center for details on how to migrate these tasks to a more recent version. See the information center library page to access the Version 5.0.2 information center.

Appendix A. Migration tools

This topic introduces the migration tools that WebSphere Application Server provides. All of the migration tools are in the `install_root/bin` directory after installation. The `WASPreUpgrade.sh` or `WASPreUpgrade.bat` scripts also ship in the `/migration/bin` directory on the product CD-ROM so that you can store the configuration of an existing release before installing the V6 product. It is important to use the migration tools for the version of Application Server that you are installing. The tools change over time. The tools on the product CD-ROM provide the necessary function for migrating from a previous release of Application Server to the one on the product CD-ROM. The tools on the CD-ROM match the product on the CD-ROM. If you use migration tools from an earlier release of Application Server, you are likely to encounter a problem with the migration.

clientUpgrade.sh (and clientUpgrade.bat)

Upgrades the client application to a new release level.

convertScriptCompatibility.sh (and convertScriptCompatibility.bat)

Used by administrators to convert their configuration from a mode that supports backward compatibility of V5.x administration scripts to a mode that is fully V6.0.

WASPreUpgrade.sh (and WASPreUpgrade.bat)

Saves the applications and configuration data from a previous installation of WebSphere Application Server to a backup directory. The `WASPostUpgrade` script restores the configuration data from the directory to the new installation. The Migration wizard calls the `WASPreUpgrade.sh` script during migration. You can also use the command to perform a manual migration, after installing the new version.

WASPostUpgrade.sh (and WASPostUpgrade.bat)

Restores the configuration data from a previous release. `WASPostUpgrade` reads the data from the backup directory where the `WASPreUpgrade` script stored the data. The Migration wizard calls the `WASPostUpgrade.sh` script during migration. You can also use the command to perform a manual migration, after installing the new version.

The clientUpgrade command

Use the **clientUpgrade** command to migrate V4.0.x and V5.x client resources to V6.0 level resources. In the process of migrating these resources, the `client-resources.xmi` file located in the client jars will be migrated to the latest level. A backup of the `client-resources.xmi` file will also be located in the client jar. If this command is not executed against the client EAR files before they are installed on V6.0, the client EARs will not operate or install correctly.

Note: WebSphere Application Server Versions 5.1 and higher do not support the Windows NT platform.

The command file is located in the `bin` subdirectory of the `WAS_install_root`, or the `ND_install_root` directory. By default, the `WAS_install_root` for WebSphere Application Server and WebSphere Application Server Enterprise is:

- Windows operating platforms - `drive\WebSphere\AppServer` directory
- AIX or UNIX-based operating platforms - `/usr/WebSphere/AppServer` directory

By default, the `ND_install_root` for WebSphere Application Server Network Deployment is:

- Windows operating platforms - `drive\WebSphere\DeploymentManager` directory
- AIX or UNIX-based operating platforms - `/usr/WebSphere/DeploymentManager` directory

Linux and UNIX-based platforms:

```
clientUpgrade.sh EAR_file [-clientJar client_jar ]  
                        [-logFileLocation logFileLocation]  
                        [-traceString trace_spec [-traceFile file_name ]]
```

Windows platforms:

```
clientUpgrade.bat EAR_file [-clientJar client_jar ]  
                        [-logFileLocation logFileLocation]  
                        [-traceString trace_spec [-traceFile file_name ]]
```

Parameters

Supported arguments include the following:

EAR_file

Use this parameter to specify the fully qualified path to the EAR file that contains client JAR files to process.

-clientJar

Use this optional parameter to specify a JAR file for processing. If not specified, the program transforms all client JAR files in the EAR file.

-logFileLocation log_file_location

Use this optional parameter to specify an alternate location to store the log output.

-traceString trace_spec -traceFile file_name

Use these optional parameters to gather trace information for IBM Service personnel. Specify a trace_spec of "*=all=enabled" (with quotation marks) to gather all trace information.

The following example demonstrates correct syntax:

```
clientUpgrade EAR_file -clientJar.ejbJarFile
```

The convertScriptCompatibility command

The **convertScriptCompatibility** command is a migration tool used by administrators for converting a configuration from a mode that supports backward compatibility of V5.x administration scripts to a mode that is fully V6.0. This command converts WebSphere Common Configuration Model (WCCM) objects of type processDef to use processDefs as defined in the 6.0 server.xml model. There can be only one occurrence of a processDefs object in a server configuration. If an existing processDefs object is found when performing this conversion, it is used and updated; otherwise a new object is created. The **convertScriptCompatibility** command also maps existing transport entries to channel support. This affects server.xml and serverindex.xml files. The values of the transport settings are used to create new channel entries.

convertScriptCompatibility.sh command syntax for Linux and UNIX-based platforms

The command syntax is as follows:

```
convertScriptCompatibility.sh [-help]  
                             [-backupConfig true | false]  
                             [-profileName profile_name]  
                             [-nodeName node_name [-serverName server_name]]  
                             [-traceString trace_spec [-traceFile file_name]]
```

convertScriptCompatibility.bat command syntax for Windows platforms

The command syntax is as follows:

```
convertScriptCompatibility.bat [-help]  
                              [-backupConfig true | false]  
                              [-profileName profile_name]  
                              [-nodeName node_name [-serverName server_name]]  
                              [-traceString trace_spec [-traceFile file_name]]
```


Parameters

Supported arguments include the following:

-help

Displays help for this command

-backupConfig true | false

An optional parameter used to back up the existing configuration of the current instance. The default is true, to use the **backupConfig** command to save a copy of the current configuration into the *profile_name/temp* directory. Use the **restoreConfig** command to restore that configuration as required.

-profileName profile_name

An optional parameter used to specify a profile configuration in the V6 environment. If not specified, the default profile will be used. If the default has not been set or cannot be found, the system will return an error.

-nodeName node_name -serverName server_name

Optional parameters used to specify a node name and a server name for the program to update. If neither is specified, all nodes and servers in the configuration are converted. When you use **-serverName** in conjunction with **-nodeName**, all processing will be limited to the specified *node_name*.

-traceString trace_spec -traceFile file_name

Optional parameters to gather trace information for IBM Service personnel. Specify a trace specification of `"*=a1l=enabled"` (with quotation marks) to gather all trace information.

WASPreUpgrade command

The **WASPreUpgrade** command is a migration tool that saves the configuration and applications of a previous version or release to a Version 6.0 node.

Location of command file

The command file is located and should be run in the `WAS60_install_root/bin` directory. The command file is a script named `WASPreUpgrade.sh` for Linux-based platforms or `WASPreUpgrade.bat` for Windows platforms.

WASPreUpgrade.sh command syntax for Linux-based platforms

The command syntax is as follows:

```
WASPreUpgrade.sh backupDirectory
                  currentWebSphereDirectory
                  [adminNodeName]
                  [-nameServiceHost host_name [-nameServicePort port_number ]]
                  [-import xmiDataFile ]
                  [-traceString trace_spec [-traceFile file_name ]]
```

WASPreUpgrade.bat command syntax for Windows platforms

```
WASPreUpgrade backupDirectory
               currentWebSphereDirectory
               [adminNodeName]
               [-nameServiceHost host_name [-nameServicePort port_number ]]
               [-import xmiDataFile ]
               [-traceString trace_spec [-traceFile file_name ]]
```

Parameters

The first two arguments are required. The third argument is required and positional only when upgrading from WebSphere Application Server Advanced Edition, Version 4.0.x.

Supported arguments include:

backupDirectory

Required parameter of the directory in which the WASPreUpgrade tool stores the saved configuration and files, and from which the WASPostUpgrade tool later reads the configuration and files. The WASPreUpgrade tool creates the directory if the directory does not already exist.

currentWebSphereDirectory

Required positional name of the installation/instance root for the current V4.x, V5.0.x, or V5.1.x installation. This version can be any form of WebSphere Application Server, V4.0.x, and any form of V5.0.x or V5.1.x, including WebSphere Application Server Express.

adminNodeName

Optional, positional name of the node containing the administrative server for the currently installed product. The value of this argument must match the node name given in the topology tree on the Topology tab of the administrative console for the currently installed product. The WASPreUpgrade tool calls the XMLConfig tool using this parameter. This parameter is only required when upgrading from WebSphere Application Server Advanced Edition, Version 4.0.x.

-nameServiceHost *host_name* **-nameServicePort** *port_number*

When specified, the WASPreUpgrade tool passes these optional parameters to the XMLConfig tool. Use these parameters to override the default host name and port number used by the XMLConfig tool. This parameter is only applicable when upgrading from WebSphere Application Server Advanced Edition, Version 4.0.x.

-import *xmiDataFile*

The name of the WebSphere Application Server Advanced Single Server Edition or Advanced Edition, Version 4.0 XML Metadata Interchange (XMI) configuration file to process. This parameter is optional because the program uses the `config\server-cfg.xml` file by default.

When migrating a configuration that uses anything other than the default `server-cfg.xml` file name, you must use the `-import` option along with a path to point to the non-default XMI configuration file. You also must use the `-import` and `path` option when running the WASPostUpgrade tool later, to point to the non-default XMI configuration file in the directory created by the WASPreUpgrade tool.

-traceString *trace_spec* **-traceFile** *file_name*

Optional parameters to gather trace information for IBM Service personnel. Specify a trace specification of `"*=all=enabled"` (with quotation marks) to gather all trace information.

Logging

The WASPreUpgrade tool displays status to the screen while it runs. The tool also saves a more extensive set of logging information in the *backup* directory. You can view the `WASPreUpgrade.log` file with a text editor.

Migrated resources

WASPreUpgrade migrates all of your resources and applications, but does not migrate entities in your `classes` directory.

Migration saves the following files in the backup directory.

For Version 5.x:

- `classes` (not saved for iSeries)
- `config`
- `installableApps`
- `installedApps`
- `properties`

WASPreUpgrade also saves all instances created in the V5.x environment. (For iSeries, WASPreUpgrade must be invoked for each instance.)

For Version 4.0.x:

- bin/setupCmdLine.sh (or bin/setupCmdLine.bat for Windows platforms)
- classes (not saved for iSeries)
- config
- installableApps
- installedApps (by default unless overridden within a specified developer configuration file)
- installedConnectors (Version 4.x Advanced Edition only)
- properties

Migrating from V4.0.x Advanced Edition

The following example specifies a backup directory named backupDirectory, and identifies the root of the existing installation as d:\WebSphere\AppServer.

```
WASPreUpgrade backupDirectory d:\WebSphere\AppServer yourNodeName
```

Migrating from V4.0.x Advanced Single Server Edition with multiple backup directories

This example shows how to migrate incrementally, to migrate separate configuration files from a single node with a single installation of WebSphere Application Server Advanced Single Server Edition. To migrate more than one configuration file, you must run the WASPreUpgrade tool multiple times to multiple backup directories because not all of the applications are in the same installedApps directory. For this reason, using a single backup directory for all runs of the WASPreUpgrade tool is not recommended. Use a separate backup directory for each run. The intent of this example is to show how to migrate a single node with multiple configuration files.

1. Run the following WASPreUpgrade commands to migrate applications A, B, C, D, and E, which reside in three separate application directories. Server assumptions include:
 - The Application Server uses the default configuration file, server-cfg.xml, as well as myServer1-cfg.xml and 01dServer-cfg.xml.

```
> WASPreUpgrade C:\WAS4ABBACKUP G:\WebSphere\AppServer
> WASPreUpgrade C:\WAS4CDBACKUP G:\WebSphere\AppServer
  -import G:\WebSphere\AppServer\config\myServer1-cfg.xml
> WASPreUpgrade C:\WAS4EBACKUP G:\WebSphere\AppServer
  -import G:\WebSphere\AppServer\config\01dServer-cfg.xml
```
2. Run the following WASPostUpgrade commands to restore the applications and configurations to the Version 6 Application Server:

```
> WASPostUpgrade C:\WAS4ABBACKUP
> WASPostUpgrade C:\WAS4CDBACKUP -import C:\WAS4CDBACKUP\myServer1-cfg.xml
> WASPostUpgrade C:\WAS4EBACKUP -import C:\WAS4EBACKUP\01dServer-cfg.xml
```

Migrating from V5.x

This example shows how to migrate a single instance of the base WebSphere Application Server, V5.x. Verify that you have stopped all Java processes related to the WebSphere Application Server product that you are migrating.

1. Run the following WASPreUpgrade.bat command to migrate all applications in the installedApps directory of the V5.x Application Server, which has an installation root of C:\Program Files\WebSphere\AppServer.

```
WASPreUpgrade "C:\WAS5xxBACKUP" C:\Program Files\WebSphere\AppServer
```
2. Run the following WASPostUpgrade commands to restore the applications and configurations to the Version 6 Application Server:

```
WASPostUpgrade "C:\WAS5xxBACKUP"
```

WASPostUpgrade command

The **WASPostUpgrade** command is a migration tool for adding the configuration and applications of a previous version or release to a Version 6.0 WebSphere Application Server node. The configuration includes migrated applications. The tool adds all migrated applications into the *install_root/installedApps* directory for the Version 6 installation. The tool retrieves the saved configuration that was created by the WASPreUpgrade tool from the *backupDirectory* that you specify.

Location of command file

The command file is located and should be run in the *WAS60_install_root/bin* directory. The command file is a script named *WASPostUpgrade.sh* for Linux-based platforms or *WASPostUpgrade.bat* for Windows platforms.

WASPostUpgrade.sh command syntax for Linux-based platforms

The command syntax is as follows:

```
WASPostUpgrade.sh backupDirectory
                    [-oldProfile profile_name]
                    [-profileName profile_name]
                    [-import xmi_data_file]
                    [-scriptCompatibility true | false]
                    [-portBlock port_starting_number]
                    [-backupConfig true | false]
                    [-replacePorts true | false]
                    [-substitute "key1=value1 [key2=value2; [...]]"]
                    [-instance instanceName -hostName hostname]
                    [-includeApps true | false]
                    [-traceString trace_spec [-traceFile file_name]]
                    [-scriptCompatibility true | false ]
                    [-connectionTimeout <timeoutInMinutes>]
```

The first argument is required.

WASPostUpgrade.bat command syntax for Windows platforms

```
WASPostUpgrade backupDirectory
                [-oldProfile profile_name]
                [-profileName profile_name]
                [-import xmi_data_file]
                [-scriptCompatibility true | false]
                [-portBlock port_starting_number]
                [-backupConfig true | false]
                [-replacePorts true | false]
                [-substitute "key1=value1 [key2=value2; [...]]"]
                [-instance instance_name -hostName host_name]
                [-includeApps true | false]
                [-traceString trace_spec [-traceFile file_name]]
                [-scriptCompatibility true | false ]
                [-connectionTimeout <timeoutInMinutes>]
```

The first argument is required.

Parameters

Supported arguments include:

backupDirectory

Required parameter of the directory in which the WASPreUpgrade tool stores the saved configuration and files. The WASPostUpgrade tool reads the configuration and files stored in this directory. The WASPreUpgrade tool creates this directory if it does not already exist.

-oldProfile *profile_name*

An optional parameter for migrating instances from previous WebSphere Application Server versions or profiles. The instance or profile must already exist in the migration backup directory before executing this command. In V5.x, unique instance names were defined by the concatenation of `-instanceName` and `-hostName`; this concatenation forms the *profile_name* that you need to use with the `-oldprofile` parameter. In V5.x, this concatenation is stored in the *install_root*\properties directory, in a file called `wsinstance.config`.

-profileName *profile_name*

An optional parameter for migrating to profiles in V6. You must have already created this profile before calling `WASPostUpgrade`. If `-profileName` is not specified, the default profile will be used. If no default profile is found, the system will report an error.

-import *xmi_data_file*

The name of the WebSphere Application Server Advanced Single Server Edition or Advanced Edition, Version 4.0 XML Metadata Interchange (XMI) configuration file to process. This parameter is optional because the program uses the `config\server-cfg.xml` file by default.

When migrating a configuration that uses anything other than the default `server-cfg.xml` file name, you must use the `-import` option along with the path to the non-default XMI configuration file in the directory created by the `WASPreUpgrade` tool.

-portBlock *port_starting_number*

An optional parameter used to specify the starting value to use when creating ports.

-backupConfig *true | false*

An optional parameter used to back up the existing configuration of the current profile before adding the saved configuration from the earlier release to the current instance. The default is `true`, to use the **backupConfig** command to save a copy of the current configuration into the *profile_name*/temp directory. Use the **restoreConfig** command to restore that configuration as required.

-replacePorts *true | false*

An optional parameter used to define how to migrate virtual host and server transport ports. The default for migrations from V4.0.x is **false** (do not replace default port definitions); the default for migrations from V5.x is **true** (do replace default port definitions). Migrating adds configuration data from the previous version to the existing data in the V6.0 configuration. In some cases existing port definitions from the earlier release were carefully set to avoid port conflicts with other products. In such cases, it is likely that you would want to migrate the settings into V6.0. Use the `-replacePorts` parameter to totally replace settings in the V6.0 environment with the settings from the previous version. Select **true** to replace all virtual host alias port settings during migration. If migrating from V5.0.x or later, transport settings in existing servers are replaced by the settings from the previous version.

-substitute *key1=value1;key2=value2;...*

Optional argument passed to the XMLConfig tool. Specify values for security variables to substitute (for example, `-substitute "NODE_NAME=admin_node;APP_SERVER=default_server"`).

In the input XML data file, each key appears as `key` for substitution. In the above example, this argument substitutes any occurrence of `$NODE_NAME$` with `admin_node` and `APP_SERVER` with `default_server` in the input XML file.

If the substitution string contains semicolons, use `$semiColon$` to distinguish the string from the ";" delimiter. On Linux platforms, add an escape character to each dollar sign (\$) within the substitution string (for example, `\$semiColon\$`).

This parameter is applicable for configurations saved from Advanced Edition, Version 4.0.x.

-instance *instance_name* **-hostName** *host_name*

Optional parameters used for migrating instances from previous versions of WebSphere Application Server. An instance of *instance_name* *host_name* must already exist in the current WebSphere Application Server configuration before this command is executed.

-includeApps true | false

An optional parameter used to specify whether to migrate only the old configuration, or to include user enterprise applications as part of the migration. The default is **true**. WebSphere Application Server system applications will migrate regardless of the value set by this parameter.

-traceString *trace_spec* -traceFile *file_name*

Optional parameters to gather trace information for IBM Service personnel. Specify a trace specification of `"*=all=enabled"` (with quotation marks) to gather all trace information.

-scriptCompatibility true | false

An optional parameter used to specify whether migration should create Transport and ProcessDef definitions in the configuration instead of Channels and ProcessDefs. Use this parameter if you have existing wsadmin scripts or programs that create or modify Transport or ProcessDef definitions. The default is true.

-connectionTimeout *timeoutInMinutes*

This is an optional parameter to be used if a SOAP/RMI timeout occurs when performing a migration of a managed node. This parameter is specified in minutes (the default value is 10). It is possible to run into a SOAP/RMI timeout issue when migrating a very large or very complex configuration; if this happens, you should increase the amount of time before a SOAP/RMI connection timeout occurs and run migration again.

Logging

The WASPostUpgrade tool displays status to the screen while running. This tool also saves a more extensive set of logging information in the logs directory. You can view the WASPostUpgrade.log file with a text editor.

Logging

The WASPostUpgrade tool displays status to the screen while running. This tool also saves a more extensive set of logging information in the logs directory. You can view the WASPostUpgrade.log file with a text editor.

Migrating from V4.0.x Advanced Edition

The following example specifies a backup directory named backupDirectory, and identifies the root of the existing installation as d:\WebSphere\AppServer.

```
WASPreUpgrade backupDirectory d:\WebSphere\AppServer yourNodeName
```

Migrating from V4.0.x Advanced Single Server Edition with multiple backup directories

This example shows how to migrate separate configuration files incrementally from a single node with a single installation of WebSphere Application Server Advanced Single Server Edition. To migrate more than one configuration file, you must run the WASPreUpgrade tool multiple times to multiple backup directories because not all of the applications are in the same installedApps directory. For this reason, using a single backup directory for all the runs of the WASPreUpgrade tool is not recommended. Use a separate backup directory for each run. The intent of this example is to show how to migrate a single node with multiple configuration files.

1. Run the following WASPreUpgrade commands to migrate applications A, B, C, D, and E, which reside in three separate application directories. Server assumptions include:

- The Application Server uses the server-cfg.xml default configuration file, as well as the myServer1-cfg.xml and the OldServer-cfg.xml files.

```
> WASPreUpgrade "C:\WAS4ABBACKUP" G:\WebSphere\AppServer
> WASPreUpgrade "C:\WAS4CDBACKUP" G:\WebSphere\AppServer
  -import G:\WebSphere\AppServer\config\myServer1-cfg.xml
> WASPreUpgrade "C:\WAS4EBACKUP" G:\WebSphere\AppServer
  -import G:\WebSphere\AppServer\config\OldServer-cfg.xml
```


2. Run the following WASPostUpgrade commands to restore the applications and configurations to the Version 6 Application Server:

```
> WASPostUpgrade C:\WAS4ABBACKUP
> WASPostUpgrade "C:\WAS4CDBACKUP" -import "C:\WAS4CDBACKUP\myServer1-cfg.xml"
> WASPostUpgrade "C:\WAS4EBACKUP" -import "C:\WAS4EBACKUP\01dServer-cfg.xml"
```

Migrating from V5.x

This example shows how to migrate a single instance of the base WebSphere Application Server, V5.x. Verify that you have stopped all Java processes related to the WebSphere Application Server product that you are migrating.

1. Run the following WASPreUpgrade.bat command to migrate all applications in the installedApps directory of the V5.x Application Server, which has an installation root of C:\Program Files\WebSphere\AppServer.

```
WASPreUpgrade "C:\WAS5xxBACKUP" C:\Program Files\WebSphere\AppServer
```

2. Run the following WASPostUpgrade commands to restore the applications and configurations to the Version 6 Application Server:

```
WASPostUpgrade "C:\WAS5xxBACKUP"
```

Using the Migration wizard

Version 6 migration always works by migrating an earlier supported version of WebSphere Application Server to a Version 6 profile.

Before using the Migration wizard, you must have access to the existing, previous version of WebSphere Application Server, and you must also have a valid Version 6 profile.

The Express product creates a valid Version 6 profile for a stand-alone Application Server during installation.

The Migration wizard is new as of Version 6. The wizard is the graphical interface to the Version 6 migration tools, the “WASPreUpgrade command” on page 137 and the “WASPostUpgrade command” on page 140, which are command line tools. This topic describes how to use the Migration wizard to preform migrations to V6.

- Migrate a V4.x Application Server to a V6 stand-alone Application Server
- Migrate a V5 Application Server to a V6 stand-alone Application Server

Migrating from V4 to V6 with the Migration wizard

Version 6 migration always works by migrating an earlier supported version of WebSphere Application Server to an Application Server *profile*.

Collect the following information about your V4 installation before you begin this procedure. The Migration wizard prompts you for the information during the migration.

Legend: Supported versions and their keys in the following table are:

- IBM WebSphere Application Server Advanced Single Server Edition, Version 4.0.2 and later (**V4.0.y AEs**)
- IBM WebSphere Application Server Advanced Edition, Version 4.0.2 and later (**V4.0.y AE**)
- IBM WebSphere Application Server Enterprise Edition, Version 4.1 and later (**V4.1.y EE**)

Table 10. V4 information that is required

Panel and field in wizard	Version 4 product			Record your value here	Link to description of field
	V4.0.y AEs	V4.0.y AE	V4.1.y EE		
Installation root directory	Yes	Yes	Yes		See “WASPreUpgrade command” on page 137 for a description of the current WebSphere Directory parameter.
Configuration file name	Optional: Defaults to config / server-cfg.xml file.	No	No, if EE is installed on top of AE. Optional if EE is installed on top of AEs: Defaults to config / server-cfg.xml file.		The name of the XML configuration file to process. See “WASPreUpgrade command” on page 137 for a description of the import parameter.
Use port values assigned to previous installation?	Optional; defaults to false.	Optional; defaults to false.	Optional; defaults to false.		Specifying “true” causes any ports of matching virtual hosts to first be removed from the new configuration before adding the new values. Specifying “false” for this parameter adds new port values. See “WASPostUpgrade command” on page 140 for a description of the replace Ports parameter.

Before using the Migration wizard, you must have access to the existing, previous version of WebSphere Application Server. You must also have a valid Version 6 profile to serve as the target of the migration.

You should have installed Version 6 already. Use the Profile creation wizard to create a new Application Server instance if the target profile for the migration does not yet exist. After verifying that a valid Version 6 target exists, you can start the procedure.

See the *Getting Started* PDF for more information.

Note: When you create the V6 profile, the NodeName you choose must match one of the NodeNames in the V4.0.x environment.

The Express product creates a valid Version 6 profile for a stand-alone Application Server during installation.

This topic describes using the Migration wizard to migrate a Version 4.x Application Server to a Version 6 stand-alone Application Server.

The Migration wizard is new as of Version 6. The wizard is the graphical interface to the Version 6 migration tools, which are the “WASPreUpgrade command” on page 137 and the “WASPostUpgrade command” on page 140. The migration tools are command line tools.




You must supply the following V6 information about the target profile during the migration:

Table 11. Version 6 information

Panel and field in wizard	Record your value here	Link to description of field
Backup directory name		See “WASPreUpgrade command” on page 137 or “WASPostUpgrade command” on page 140 for a description of the backup Directory Name parameter.
Profile name		See “WASPostUpgrade command” on page 140 for a description of the profile Name parameter.

After gathering all of the information that is required during the migration, use the wizard to migrate a Version 4.x Application Server to a Version 6 stand-alone Application Server.

1. Start the First steps console.

-   `.install_root/profiles/profile_name/firststeps/firststeps.sh`
-  `install_root\profiles\profile_name\firststeps\firststeps.bat`

2. Click the **Migration wizard** option on the First steps console.

The First steps console starts the Migration wizard. The wizard displays its Welcome panel.

3. Read the Welcome panel to learn about the migration, then click **Next**.

The wizard displays the Detected versions of WebSphere Application Server panel.

4. Select a previous version, then click **Next** to continue.

Select the check box to specify the location of a previous installation that might not appear in the list. **Version 4.0.y AEs** requires you to identify the location of the configuration file. You do not need to identify the location of the configuration file for other V4 editions.

The default location to use for the previous installation varies depending on the V4.x product that you have installed. The following list shows default locations:

- IBM WebSphere Application Server Advanced Single Server Edition, Version 4.0.2 and later (**V4.0.y AEs**) uses a default location for the `config\server-cfg.xml` file.
- IBM WebSphere Application Server Advanced Edition, Version 4.0.2 and later (**V4.0.y AE**) uses the output from the XMLConfig tool. You must have the Advanced Edition administrative server running during the migration.
- IBM WebSphere Application Server Enterprise Edition, Version 4.1 and later (**V4.1.y EE**) uses the location of the product that it is extending, which can be either AEs or AE.

If the previous version specification is correct, the wizard displays the Migration backup directory panel.

5. Specify a backup directory, then click **Next** to continue.

If the directory that you specify already has files in it, the wizard displays a warning message. Specify an empty directory or a directory that does not yet exist.

If the V6 profile is valid, the wizard displays the Target profile selection panel.

6. Select the target profile from the list of valid profiles for the installation, then click **Next**.

Use the Profile creation wizard to create a target profile for the migration if one does not yet exist.

The wizard displays the Port value assignment panel.

7. Choose whether to use port values assigned to the previous installation, or generate new V6 port values that are already assigned to the target profile, then click **Next**.

The wizard begins the premigration. The wizard displays a Migration status panel during the premigration.

If the premigration is not successful, the wizard displays a failure panel. If the premigration is partially successful, the wizard displays a warning panel.

Correct any problems and retry the migration.

If the premigration is successful, the wizard displays an indication of success.

8. Click **Next** to start the postmigration.

If the postmigration is not successful, the wizard displays a failure panel. If the postmigration is partially successful, the wizard displays a warning panel.

Correct any problems and retry the migration.

If the postmigration is successful, the wizard displays an indication of success.

9. Click **Finish** to exit the migration wizard.

Migrating a V5 Application Server to a V6 Stand-alone Application Server with the Migration wizard

Version 6 migration always works by migrating an earlier supported version of WebSphere Application Server to an Application Server *profile*.

Collect the following information about your V5 installation before you begin this procedure. The Migration wizard prompts you for the information during the migration:

- Installation root directory
See “WASPreUpgrade command” on page 137 for a description of the `currentWebSphereDirectory` parameter.
- Use port values assigned to previous installation?
Specifying “true” causes any ports of matching `VirtualHosts` to first be removed from the new configuration before adding the new values.
Specifying “false” for this parameter will just add new port values.
See “WASPostUpgrade command” on page 140 for a description of the `replacePorts` parameter.
- Backup directory name
See “WASPreUpgrade command” on page 137 for a description of the `backupDirectory` parameter.
- Target profile name.
See “WASPostUpgrade command” on page 140 for a description of the `profileName` parameter.

Before using the Migration wizard, you should have installed Version 6 already. Use the Profile creation wizard to create a valid new Application Server profile if one does not already exist. After verifying that a valid Version 6 target exists, you can start the procedure.




The Express product creates a valid Version 6 profile for a stand-alone Application Server during installation.

This topic describes using the Migration wizard to migrate a Version 5.x Application Server to a Version 6 stand-alone Application Server.

The Migration wizard is new as of Version 6. The wizard is the graphical interface to the Version 6 migration tools, which are the “WASPreUpgrade command” on page 137 and the “WASPostUpgrade command” on page 140. The migration tools are command-line tools.

Use the wizard to migrate a Version 5.x Application Server to a Version 6 stand-alone Application Server.

1. Start the First steps console.

-   `.install_root/profiles/profile_name/firststeps/firststeps.sh`
-  `install_root\profiles\profile_name\firststeps\firststeps.bat`

2. Click the **Migration wizard** option on the First steps console.

The First steps console starts the Migration wizard. The wizard displays its Welcome panel.

3. Read the Welcome panel to learn about the migration, then click **Next**.

The wizard displays the Detected versions of WebSphere Application Server panel.

4. Select a previous version, then click **Next** to continue.

Select the check box to specify the location of a previous installation that might not appear in the list.

If the previous version specification is correct, the wizard displays the Target profile selection panel.

5. Select the target profile from the list of valid V6 profiles for the installation, then click **Next**.

Use the Profile creation wizard to create a target profile for the migration if one does not yet exist.

The wizard displays the Port value assignment panel.

6. Choose whether to use port values assigned to the previous installation, or generate new V6 port values that are already assigned to the target profile, then click **Next**.

The wizard begins the premigration. The wizard displays a Migration status panel during the premigration.

If the premigration is not successful, the wizard displays a failure panel. If the premigration is partially successful, the wizard displays a warning panel.

Correct any problems and retry the migration.

If the premigration is successful, the wizard displays an indication of success.

7. Click **Next** to start the postmigration.

If the postmigration is not successful, the wizard displays a failure panel. If the postmigration is partially successful, the wizard displays a warning panel.

Correct any problems and retry the migration.

If the postmigration is successful, the wizard displays an indication of success.

8. Click **Finish** to exit the migration wizard.

Appendix B. Troubleshooting migration

To resolve problems encountered in migrating from an older version of WebSphere Application Server, determine whether your problems occur using the pre-upgrade tool or the post-upgrade tool.

- **The following problems can occur when using the pre-upgrade tool:**

1. A "Not found" or "no such file or directory" message is returned from the WASPostUpgrade or WASPreUpgrade tool. This problem can occur if you are trying to run the WASPostUpgrade tool or the WASPreUpgrade tool from a directory other than *install_dir\bin*. Verify that the WASPostUpgrade or WASPreUpgrade .bat or .sh files reside in the *install_dir\bin* directory, and launch either file from that location.
1. The DB2 JDBC driver and DB2 JDBC driver (XA) cannot be found in the drop down list of supported JDBC providers in the administrative console. The administrative console no longer displays deprecated JDBC provider names. The new JDBC provider names used in the administrative console are more descriptive and less confusing. The new providers will differ only by name from the deprecated ones.

The deprecated names will continue to exist in the *jdbc-resource-provider-templates.xml* file for migration reasons (for example, for existing JAAS scripts). However, you are encouraged to use the new JDBC provider names in your JAAS scripts.

2. You receive message: MIGR0108E: The specified WebSphere directory does not contain WebSphere version that can be upgraded.

Several possible reasons for this error exist:

- If WebSphere Application Server version 4.0.x is installed, you might not have run the WASPreUpgrade tool from the *bin* directory of the Version 5 installation root.
 - a. Look for the following message to display when the WASPreUpgrade tool runs: IBM WebSphere Application Server, Release 4.0. This message indicates that you are running the WebSphere Application Server Release 4.0 migration utility, not the version 5 migration utility.
 - b. Alter your environment path or change the current directory so that you can launch the WebSphere Application Server Version 5 WASPreUpgrade program.
- WebSphere Application Server version 5 might be installed in the same root directory as the earlier version.
 - a. Browse the directory structure of the older version to see whether it contains new Version 5.0 directories, such as *WebSphere\AppServer\logs\ffdc*.
 - b. Uninstall all versions of WebSphere Application Server.
 - c. Reinstall and reconfigure the older version.
 - d. Install WebSphere Application Server Version 5 in a different root directory than the previous one.
- An invalid directory might have been specified when launching the WASPostUpgrade tool, or the WASPreUpgrade tool has not been run.

3. MIGR0125E: The call to XMLConfig was not successful error when trying to run WASPreUpgrade. The WASPreUpgrade tool saves selected files from WebSphere Application Server Version 3.5.x and Version 4.x *bin* directories. It also exports the existing application server configuration from the repository.

If you are migrating from WebSphere Application Server Version 4.0.x Advanced Edition, the WASPreUpgrade command calls the **XMLConfig** command to export the existing application server configuration from the repository. If errors occur during this part of the WASPreUpgrade command, you might have to apply fixes to the installation to successfully complete the export step. Contact IBM Support for the latest applicable fixes using the link at the end of this topic.

- **The following problems can occur when using the post-upgrade tool:**

1. A "Not found" or "no such file or directory" message is returned from the WASPostUpgrade or WASPreUpgrade tool. This problem can occur if you are trying to run the WASPostUpgrade tool or

the WASPreUpgrade tool from a directory other than *install_dir\bin*. Verify that the WASPostUpgrade or WASPreUpgrade .bat or .sh files reside in the *install_dir\bin* directory, and launch either file from that location.

2. You receive message: MIGR0102E: Invalid Command Line. MIGR0105E: You must specify the primary node name. The most likely cause of this error is that Version 4.0.x of the WebSphere Application Server is installed, and the WASPostUpgrade tool was not run from the bin directory of the WebSphere Application Server Version 5 installation root.

If the following messages appear when the WASPostUpgrade tool is run, the Version 4.0 migration tool was run:

- IBM WebSphere Application Server, Release 4.0
- MIGR0002I: java com.ibm.websphere.migration.postupgrade.WASPostUpgrade <backupDirectoryName> -adminNodeName <primary node name> [-nameServiceHost <hostName> [-nameServicePort <portNumber>]] [-substitute <"key1=value1[;key2=value2;...]">] In input xml file, the key(s) should appear as \$key\$ for substitution.) [-import <xml data file>] [-traceString <trace specification> [-traceFile <filename>]]}"

To correct this problem, run the WASPostUpgrade command from the bin directory of the WebSphere Application Server Version 5 installation root.

3. You receive message: MIGR0108E: The specified WebSphere directory does not contain WebSphere version that can be upgraded.

Several possible reasons for this error exist:

- If WebSphere Application Server version 4.0.x is installed, you might not have run the WASPreUpgrade tool from the bin directory of the Version 5 installation root.
 - a. Look for the following message to display when the WASPreUpgrade tool runs: IBM WebSphere Application Server, Release 4.0. This message indicates that you are running the WebSphere Application Server Release 4.0 migration utility, not the version 5 migration utility.
 - b. Alter your environment path or change the current directory so that you can launch the WebSphere Application Server Version 5 WASPreUpgrade program.
 - WebSphere Application Server version 5 might be installed in the same root directory as the earlier version.
 - a. Browse the directory structure of the older version to see whether it contains new Version 5.0 directories, such as WebSphere\AppServer\logs\ffdc.
 - b. Uninstall all versions of WebSphere Application Server.
 - c. Reinstall and reconfigure the older version.
 - d. Install WebSphere Application Server Version 5 in a different root directory than the previous one.
 - An invalid directory might have been specified when launching the WASPostUpgrade tool, or the WASPreUpgrade tool has not been run.
4. You receive message: MIGR0116E: The backup directory [migration_backup_directory] does not contain the required xml data file.

If Version 4.0.x of WebSphere Application Server is installed, you might not have run the WASPostUpgrade tool from the bin directory of the Version 5.0 installation root.

- If the message IBM WebSphere Application Server, Release 4.0 displays when launching the WASPostUpgrade program, then the wrong version of the program is executing.
 - Run the WASPostUpgrade command from the bin directory of the Version 5.0 installation root.
5. You receive error: MIGR0253E: The backup directory migration_backup_directory does not exist.

Several possible reasons for this error exist:

- The WASPreUpgrade tool was not run prior to the WASPostUpgrade tool.
 - a. Check to see if the backup directory specified in the error message exists.
 - b. If not, run the WASPreUpgrade .bat or .sh file.
 - c. Retry the WASPostUpgrade tool.

- An invalid backup directory might be specified. For example, the directory might have been a subdirectory of the V3.5.x or V4.0.x tree, which was deleted after the WASPreUpgrade tool was run and the older version of the product was uninstalled, but before the WASPostUpgrade tool was run.
 - a. Determine if the full directory structure specified in the error message exists.
 - b. If possible, rerun the WASPreUpgrade tool, specifying the correct full migration backup directory.
 - c. If the backup directory does not exist, and the older version it came from is gone, rebuild the older version from a backup repository or XML configuration file.
 - d. Rerun the WASPreUpgrade tool.
- 6. You decide that you need to run WASPreUpgrade again after you have already run WASPostUpgrade. During the course of a deployment manager or a managed node migration, WASPostUpgrade disables the old environment. If, after running WASPostUpgrade, you want to run WASPreUpgrade again against the old installation, you must execute the migrationDisablementReversal.jacl script located in the old *install_root/bin* directory. After executing this .jacl script, your V5.x environment will be in a valid state again, allowing you to run WASPreUpgrade to produce valid results.
- 7. The SOAP/RMI call up to the deployment manager times out. During the course of a federated migration, the migration process makes a call up to the DeploymentManager to perform a portion of the migration. If, after seeing the message **MIGR0388I**, you see a SOAP/RMI connection timeout exception, rerun WASPostUpgrade specifying the "-connectionTimeout" to some value greater than the default (the default is 10 minutes). A good rule of thumb is to double the value and execute WASPostUpgrade again.
- 8. A federated migration fails with message **MIGR0405E**. The migration that has taken place on your deployment manager as part of your federated migration has failed. For a more detailed reason of why this error has occurred, open the folder *yourNodeName_migration_temp* located on your deployment manager node under the ...DeploymentManagerProfile/temp directory. Example:
`/websphere60/appserver/profiles/dm_profile/temp/nodeX_migration_temp`

The logs, traces, and everything else involved in the migration for this node on the deployment manager node are located in this folder. This folder will also be required for IBM support related to this scenario.
- 9. V6 applications are lost during migration. During a federated migration, if any of the V6 applications fail to install, they will be lost during the syncing of the configurations. The reason this happens is that one of the final steps of WASPostUpgrade is to execute a syncNode command. This has the result of downloading the configuration on the deployment manager node and overwriting the configuration on the federated node. If the applications fail to install, they will not be in the configuration located on the deployment manager node. To resolve this issue, manually install the applications after migration. If they are standard V6 applications, they will be located in the *WAS_ROOT/installableApps* directory.
- **If none of these errors describes your problem:**
 1. For general tips on migration problems, see Troubleshooting the migration utility.
 2. Review the migration topic and its subtopics, which address migrating specific kinds of components.
 3. For other kinds of migration problems, such as an application imported from another version of WebSphere Application Server that will not start, look up the related problem in the *Troubleshooting and Support* PDF.
 4. Check to see if the problem has been identified and documented by looking at the IBM Support page.
 5. IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

If you did not find your problem listed, contact IBM support.

Migration utility troubleshooting tips

If you encounter problems migrating an application from a previous version of WebSphere Application Server to Version 6.0:

- Look for these log files and browse them for clues:
 - `install_dir/profile/profile_name/logs/WASPostUpgrade.time stamp.log`
 - `install_dir/profile/profile_name/logs/WASPostUpgrade.trace`
 - `migration_backup_dir/WASPreUpgrade.time stamp.log`
 - `install_dir/logs/clientupgrade.time stamp.log`
- Look for **MIGR0259I: The migration has successfully completed.** or **MIGR0271W: The migration completed with warnings.** in the `migration_backup_dir/WASPreUpgrade.time stamp.log`, `migration_backup_dir/WASPreUpgrade.time stamp.log`, or `install_dir/logs/clientupgrade.time stamp.log`.
If **MIGR0286E: The migration failed to complete.** appears, attempt to correct any problems based on the error messages that appear in the log file. After correcting any errors, rerun the command from the bin directory of the product installation root. If the errors persist, rerun the command with trace enabled.
- Look at your trace output to see more detailed message information. To locate the trace output:
 - For the WASPreUpgrade command, look in the `backupDirectory`.
 - For all other commands, look in the logs directory of the profile being modified. The names of the log files follow this format:
`commandname.timestamp.trace`

where `commandname` is WASPostUpgrade or another command, and `timestamp` is a date timestamp.

If you are still unable to find your trace output, see the *Troubleshooting and Support* PDF for more information.

See the following articles for more information:

- “WASPreUpgrade command” on page 137
- “WASPostUpgrade command” on page 140
- “The clientUpgrade command” on page 135
- Open the log analyzer on the service log of the server which is hosting the resource you are trying to access and use it to browse error and warning messages.
- With WebSphere Application Server running, run the **dumpNameSpace** on Windows or **dumpNameSpace.sh** command on Unix, and pipe, redirect, or “more” the output so that it can be easily viewed. This command results in a display of all objects in WebSphere Application Server’s namespace, including the directory path and object name.
- If the object a client needs to access does not appear, use the administrative console to verify that:
 - The server hosting the target resource is started.
 - The web module or EJB container hosting the target resource is running.
 - The JNDI name of the target resource is properly specified.
- To view detailed information on the runtime behavior of WebSphere Application Server’s Naming service, enable trace on the following components and review the output:
 - `com.ibm.ws.naming.*`
 - `com.ibm.websphere.naming.*`

If none of these steps solves the problem, see Appendix B, “Troubleshooting migration,” on page 149 for tips on specific migration problems.

Check to see if the problem is identified and documented using the links in the *Troubleshooting and Support* PDF.

If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

See the *Troubleshooting and Support* PDF for more information.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Migrating a previously non-root configuration to root

This topic describes how to run an Express server as root in WebSphere Application Server Version 6 when you previously used a non-root user ID on Linux and UNIX platforms in Version 5.x. In this scenario, the following step must be completed for the Express servers to run correctly:

Set the ownership of the profile or directory to be the same as the user under which WebSphere Application Server - Express is to run. As a root user, run the following commands:

```
cd $WASROOT/profiles/  
chown -R wasadmin default/  
chgrp -R wasgrp default/  
chmod -R 755 default/  
...  
cd $WASROOT/profiles/default/bin  
./startManager.sh
```

where the profile name is `default` run as a user in `wasadmin` with the primary group `wasgrp`.

Appendix C. Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594 USA

Appendix D. Trademarks and service marks

For trademark attribution, visit the IBM Terms of Use Web site (<http://www.ibm.com/legal/us/>).