



Migrating, coexisting, and interoperating

Note

Before using this information, be sure to read the general information under “Notices” on page 153.

Compilation date: April 20, 2006

© Copyright International Business Machines Corporation 2006. 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	v
Chapter 1. Overview and new features for migrating, coexisting, and interoperating	1
Deprecated and removed features.	1
Deprecation list.	1
Removal list	11
Chapter 2. How do I migrate, coexist, and interoperate?	17
Chapter 3. Migrating applications	19
Web applications.	19
Migrating to Java 2 Standard Edition (J2SE) 5.	19
Migrating servers from multi-broker replication domains to data replication domains	21
JavaServer Pages migration best practices and considerations.	25
JavaServer Faces	26
Migrating Web application components from WebSphere Application Server Version 5.x	30
HTTP session migration	31
EJB applications	31
Migrating enterprise bean code to the supported specification	31
Container interoperability.	35
Web services	37
Web Services-Interoperability Basic Profile	37
Migrating Web services	39
Migrating the UDDI registry	43
Migrating to Version 3 of the UDDI registry	44
Data access resources	47
Migrating applications to use data sources of the current J2EE Connector Architecture (JCA)	47
Verifying the Cloudscape v10.1.x automatic migration	50
Upgrading Cloudscape manually	53
Security	56
Migrating, coexisting, and interoperating – Security considerations	56
Enabling embedded Tivoli Access Manager	70
Propagating security policy of installed applications to a JACC provider using wsadmin scripting	71
Naming and directory	72
JNDI interoperability considerations	72
Application profiling	73
Running Version 5 Application Profiles on Version 6.	73
Application profiling interoperability	75
Asynchronous beans	76
Interoperating with asynchronous beans	76
Chapter 4. Migrating and coexisting	77
Overview of migration, coexistence, and interoperability	77
Premigration considerations.	81
API and specification migration	84
Programming model extension migration	86
Chapter 5. Migrating product configurations	87
Configuration mapping during product-configuration migration	87
Planning to migrate product configurations	91
Migrating stand-alone application servers.	91
Preparing to migrate a stand-alone application server	92
Customization Dialog walk-through for migrating a stand-alone application server	94

Checklist of activities for migrating a stand-alone application server	100
Migrating a stand-alone application server	101
Migrating Network Deployment configurations	104
Preparing to migrate a Network Deployment configuration	104
Migrating deployment managers	106
Migrating federated nodes	116
Using the migration tools to migrate product configurations.	127
The clientUpgrade command	128
convertScriptCompatibility command	129
Migrating Cloudscape databases	130
Rolling back your environment	131
Rolling back a deployment cell	131
Rolling back a managed node	132
Chapter 6. Migrating Web server configurations	135
Chapter 7. Migrating administrative scripts	137
Migrating administrative scripts from 5.x to 6.x	137
Making changes from 5.x to 6.x to accommodate changes without scripting support	137
Example: Migrating - Allowing configuration overwrite when saving a configuration	138
Migrating administrative scripts from V6.0.2 to V6.1	139
Chapter 8. Coexisting	141
Coexistence support	141
Chapter 9. Interoperating	145
Chapter 10. Configuring ports.	147
Chapter 11. Troubleshooting migration	149
Appendix. Directory conventions	151
Notices	153
Trademarks and service marks	155

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.

Chapter 1. Overview and new features for migrating, coexisting, and interoperating

Use the links provided in this topic to learn about the migration, coexistence, and interoperability features.

“Overview of migration, coexistence, and interoperability” on page 77

This topic describes common topologies that you can install with the product.

Deprecated and removed features

Various features have been either deprecated or removed in WebSphere Application Server product offerings since Version 5.0.

If a feature is listed here as deprecated, IBM intends to remove this capability in a future release of the product. Generally, IBM will not remove a feature until at least two major releases or three full years (whichever is longer) after the release in which it is deprecated. For example, features that were deprecated in WebSphere Application Server Version 5.0, Version 5.0.1, or Version 5.0.2 will not be removed from the product until after Version 6.0 because both Version 5.0.x and Version 5.1.x are considered to be major releases. In rare cases, it might become necessary to remove features sooner; such cases are indicated clearly and explicitly in the descriptions of these deprecated features in this article.

Deprecated and removed features

See the following topics to find more information on deprecated and removed features:

- “Deprecation list”
- “Removal list” on page 11

Deprecation list

The following tables summarize deprecated features by version and release. The tables indicate what is being deprecated, such as application programming interfaces (APIs), scripting interfaces, tools, wizards, publicly exposed configuration data, naming identifiers, and constants. Where possible, the tables also indicate the recommended migration action.

This article contains the following deprecation tables:

- “Features deprecated in Version 6.1”
- “Features deprecated in Version 6.0.2” on page 4
- “Features deprecated in Version 6.0.1” on page 5
- “Features deprecated in Version 6.0” on page 5
- “Features deprecated in Version 5.1.1” on page 7
- “Features deprecated in Version 5.1” on page 7
- “Features deprecated in Version 5.0.2” on page 10
- “Features deprecated in Version 5.0.1” on page 11
- “Features deprecated in Version 5.0” on page 11

Features deprecated in Version 6.1

Application Programming Model Features	
Deprecation	Recommended Migration Action

The setDatabaseDefaultIsolationLevel(int) method in the com.ibm.websphere.rsadapter.DataStoreHelperMetaData class is deprecated.	Start using the following method instead: <pre>public final void setDatabaseDefaultIsolationLevel (int helperDefaultLevel, int cusDefinedWasDefaultIsoLevel)</pre>
The following class and interface in the Mediation Framework runtime are deprecated: <ul style="list-style-type: none"> com.ibm.websphere.sib.mediation.handler.SIMessageContextException class com.ibm.websphere.sib.mediation.messagecontext.SIMediationBean MessageContext interface 	Replace all uses of the com.ibm.websphere.sib.,mediation.handler.SIMessageContextException class with the com.ibm.websphere.sib.mediation.handler.MessageContextException class. Replace all uses of the com.ibm.websphere.sib.mediation.messagecontext.SIMediationBean MessageContext interface with an equivalent interface. WebSphere Application Server does not provide an implementation of this interface.
Support for HTTP transport configuration is deprecated.	Begin moving to channel-based transport.
The following WebContainer MBean functions are deprecated: <ul style="list-style-type: none"> startTransports stopTransports restartWebApplication 	Begin moving to the channel framework. The channel framework provides the TransportChannelService MBean, which is more flexible and has more methods than the current WebContainer transport-related methods.
Support for deploying container-managed entity beans to a generic SQL database is deprecated.	If an application uses SQL92 or SQL99 because the application has to run with different relational databases, use the IBM tooling to generate deployed code for each database vendor or version that the application might use. At installation time, specify the database vendor or version that will be used with WebSphere Application Server.
The IBM WebSphere Studio tools runtime support provided by the following classes (which were used to leverage VJava tooling) is deprecated: <ul style="list-style-type: none"> com.ibm.webtools.runtime.AbstractStudioServlet com.ibm.webtools.runtime.BuildNumber com.ibm.webtools.runtime.NoDataException com.ibm.webtools.runtime.StudioPervasiveServlet com.ibm.webtools.runtime.TransactionFailureException com.ibm.webtools.runtime.WSUtilities 	Rearchitect your applications to use standard J2EE coding conventions.
The CUSTOM_HELPER constant field in the com.ibm.websphere.rsadapter.DataStoreHelper class API is deprecated.	If you create your own DataStoreHelper implementation class, do not invoke setHelperType(DataStoreHelper.CUSTOM_HELPER). Instead, let the HelperType value be set by the implementation class from which it inherits.
J2EE Resource Features	
Deprecation	Recommended Migration Action
Support for the ability to connect from either an application server or a J2EE application client to the JMS Server component of the embedded messaging feature in WebSphere Application Server Version 5 is deprecated. This deprecation includes the following: <ul style="list-style-type: none"> Ability to define JMS resource definitions for the Version 5 default messaging provider Ability to establish connections from client applications that are either running in a Version 5 environment or utilizing Version 5 default messaging provider resource definitions 	Perform the following actions: <ol style="list-style-type: none"> Ensure that any JMS Server messaging providers that are hosted on WebSphere Application Server Version 5.1 application servers are moved onto Version 6.0 or later application servers. This task is handled automatically when you migrate a Version 5.x server to Version 6.0 or later. Change all JMS resource definitions to use the new Version 6 default messaging provider instead of the Version 5 default messaging provider.
System Administration Features	
Deprecation	Recommended Migration Action
WebSphere administrative (wsadmin) scripting support for the Jacl language is deprecated.	Begin converting your existing Jacl syntax wsadmin scripts into Jython syntax wsadmin scripts, and use Jython syntax for any new wsadmin scripting. For more information, go to Application Server Toolkit > Administering applications > Developing automation scripts in this information center.
The Customization Dialog, the set of Interactive System Productivity Facility (ISPF) panels used to generate jobs and instructions for configuring and migrating the WebSphere Application Server for z/OS environment, is being deprecated.	Begin using the Profile Management tool for z/OS to generate the jobs and instructions for creating profiles.
The clientUpgrade command is deprecated.	No migration action is necessary.

The Cloudscape datastore helper (com.ibm.websphere.rsadapter.CloudscapeDataStoreHelper) and Cloudscape Network Server datastore helper (com.ibm.websphere.rsadapter.CloudscapeNetworkServerDataStoreHelper) as well as their types in DataStoreHelper are deprecated.	For existing configurations, no migration action is necessary. The migration utility changes the deprecated Cloudscape helpers to Derby helpers. For new configurations, use the Derby datastore helpers and types instead of the Cloudscape datastore helpers.
The Derby Network Server Using Universal JDBC Driver Provider, which uses the db2jcc.jar file, is deprecated.	Start using the Derby Network Server Using Derby Client provider, which uses the derbyclient.jar file and has the benefit of supporting XA.
The DB2 Legacy CLI-based Type 2 JDBC Driver provider is deprecated.	Start using the DB2 Universal JDBC Driver Provider.
Logical pool distribution support (com.ibm.websphere.csi.ThreadPoolStrategy.LogicalPoolDistribution) is deprecated.	No migration action is necessary. When this function is removed, however, all custom ORB properties that you specified for it will be ignored. The custom ORB properties of interest are com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.*.
ORB thread pool configuration as part of the Server object in the server.xml file is deprecated.	Use the thread pool configuration that is part of the ServerIndex object in the serverindex.xml file.
The transaction class mapping file that is pointed to by the configuration variable protocol_http_transport_class_mapping_file is deprecated.	Use the XML file that is pointed to by the configuration variable wlm_classification_file to map HTTP requests to WLM transaction classes. The deprecated mapping file supported specifying multiple transaction classes per row. This allowed the creation of an artificial round-robin work-dispatch effect. Although the XML file pointed to by wlm_classification_file does not support specifying multiple transaction classes per row, you can better achieve this same effect by using zWLM's round-robin option. Select this option by specifying wlm_stateful_session_placement_on=1.
The following configuration variables are deprecated: <ul style="list-style-type: none"> • protocol_http_backlog • protocol_https_backlog • protocol_iiop_backlog • protocol_iiop_backlog_ssl 	Use the HTTP transport channel listenBacklog custom property.
The SSL certificate mapping file specified by the protocol_https_cert_mapping_file environment variable is deprecated. The following related configuration variables are also deprecated: <ul style="list-style-type: none"> • protocol_https_cert_mapping_file • protocol_https_default_cert_label 	No migration action is necessary at this time. In a future version of WebSphere Application Server for z/OS, a new configuration mechanism will be provided that will allow you to choose a different SSL server certificate to be used for the SSL handshake depending on the server IP address of the socket connection. This new administrative mechanism will replace the current file format of the file pointed to by the protocol_http_cert_mapping_file variable.
The JVM system property com.ibm.websphere.sendredirect.compatibility is deprecated.	Begin modifying your applications to redirect non-relative URLs, those starting with a forward slash ("/"), relative to the servlet container root (<i>web_server_root</i>) instead of the web application context root. See the <i>Java Servlet 2.4 Specification</i> , available for download at http://jcp.org/aboutJava/communityprocess/final/jsr154/ , for information on how sendRedirect should behave.
The Web container PageList Servlet custom extension is deprecated, including the following classes: <ul style="list-style-type: none"> • com.ibm.servlet.ClientList • com.ibm.servlet.ClientListElement • com.ibm.servlet.MLNotFoundException • com.ibm.servlet.PageListServlet • com.ibm.servlet.PageNotFoundException 	Rearchitect your applications to use javax.servlet.filter classes rather than com.ibm.servlet classes. Starting with the Java Servlet 2.3 specification, javax.servlet.filter classes give you the capability to intercept requests and examine responses. They also allow provide chaining functionality as well as functionality for embellishing or truncating responses.
The following datasource custom properties are deprecated: <ul style="list-style-type: none"> • dbFailOverEnabled • connRetriesDuringDBFailover • connRetryIntervalDuringDBFailover 	Replace the deprecated properties with new properties. <ul style="list-style-type: none"> • Use validateNewConnection instead of dbFailOverEnabled. • Use validateNewConnectionRetryCount instead of connRetriesDuringDBFailover. • Use validateNewConnectionRetryInterval instead of connRetryIntervalDuringDBFailover. Note: If both properties coexist, the new properties take precedence.
Peer restart and recovery (PRR)	For transaction recovery, migrate from using PRR functionality to using high availability manager functionality.
Security Features	
Deprecation	Recommended Migration Action
The Simple WebSphere Authentication Mechanism (SWAM) is deprecated.	Use the Lightweight Third Party Authentication (LTPA) mechanism.

The LoginHelper CORBA authentication helper function (com.ibm.ws.security.util.LoginHelper) is deprecated.	Migrate to the Java Authentication and Authorization Service (JAAS) programming model. For information on this migration, see “Migrating Common Object Request Broker Architecture programmatic login to Java Authentication and Authorization Service (CORBA and JAAS)” on page 64.
z/OS System SSL support for all server types except Daemon is deprecated.	Begin converting existing wsadmin scripts that use z/OS System SSL security settings to scripts that use JSSE security settings.
Problem Determination Features	
Deprecation	Recommended Migration Action
The message ID format that is used in WebSphere Application Server Version 6.0.x and earlier is deprecated.	Use the convertlog command and the MessageConverter class to assist you in migrating tools that rely on the older message format.
The message prefixes for log files were not previously registered with the primary message registry. WebSphere Application Server Version 6.1 and later use compliant message prefixes in the output logs.	
<p>The com.ibm.etools.logging.util plug-in, the logutil.jar file, is deprecated.</p> <ul style="list-style-type: none"> Logging facility Logging facility used for logging Java primitives and complex objects to named loggers; configurable with predefined filtering levels, Logging Agent and file sinks, and output formats through an API, Eclipse plug-in manifest, or Eclipse preference panel Logging agent XML-based messaging agent used in conjunction with the IBM Agent Controller to write log and trace XML records to a logging service remotely attachable through an API or Test and Performance Tools Platform (TPTP), formally Hyades, Eclipse workbench Problem determination artifacts and messages Original implementation of the Manageability (M12) Model Problem Determination Architecture Version 1.5 and Problem Determination Artifacts Common Data Model specification used for capturing and encoding log and trace data Distributed correlator service (DCS) Distributed correlator service that is used for instrumenting correlation identifiers for correlating log and trace data across one or more hosts Java client bindings Java client bindings used to communicate with the IBM Agent Controller to launch local and remote processes, attach to running processes, and monitor active agents in a secure client environment 	<p>Begin moving plug-ins and application code using configuration files, classes, methods, or variables in the com.ibm.etools.logging.util plug-in to the following replacements:</p> <ul style="list-style-type: none"> Logging facility Replacement: Java Logging APIs in Java Version 1.4.0+; Logging Agent support for the Java Logging APIs provided in TPTP and Common Logging (com.ibm.etools.common.logging/logging.jar) Logging agent Replacement: TPTP Logging Agent (org.eclipse.hyades.logging.core/hlcore.jar) Problem determination artifacts and messages Replacement: Common Base Event Version 1.0.1 specification and TPTP implementation (org.eclipse.hyades.logging.core/hlcb101.jar) Distributed correlator service (DCS) Replacement: TPTP Correlation Service (org.eclipse.hyades.execution.correlation/hcorrelation.jar) Java client bindings Replacement: TPTP Java Client Bindings (org.eclipse.hyades.execution/hexl.jar) <p>For more information, see the com.ibm.etools.logging.util/doc\IBM_Logging_Utilities_Migration_Guide.html document.</p>

Features deprecated in Version 6.0.2

Application Programming Model Features	
Deprecation	Recommended Migration Action
<p>The following methods from class com.ibm.websphere.runtime.ServerName are deprecated:</p> <pre>initialize(java.lang.String cell, java.lang.String node, java.lang.String server) was390Initialize(byte[] a_stoken, String a_printable_stoken, String a_jsabpref, int a_pid, int an_asid, String a_jsabjbnm) was390Initialize(byte[] a_stoken, java.lang.String a_printable_stoken, java.lang.String a_jsabpref, int a_pid, int an_asid, java.lang.String a_jsabjbnm, java.lang.String a_smcasid)</pre>	<p>These methods are for WebSphere Application Server runtime use only. Applications should not call these methods.</p>
Performance Features	
Deprecation	Recommended Migration Action
The com.ibm.websphere.cache.DistributedLockingMap interface is deprecated.	Do not use the com.ibm.websphere.cache.DistributedLockingMap interface because this interface is not supported by the WebSphere Application Server runtime.

The TYPE_DISTRIBUTED_LOCKING_MAP constant that is defined in the com.ibm.websphere.cache.DistributedObjectCache class is deprecated.	Do not use the TYPE_DISTRIBUTED_LOCKING_MAP constant that is defined in the com.ibm.websphere.cache.DistributedObjectCache class because this constant is not supported by the WebSphere Application Server runtime.
--	--

Features deprecated in Version 6.0.1

<u>Security Features</u>	
Deprecation	Recommended Migration Action
The z/OS Secure Authentication Service (z/SAS) IOP security protocol is deprecated.	Use the Common Secure Interoperability Version 2 (CSIV2) protocols.

Features deprecated in Version 6.0

<u>Application Programming Model and Container Support Features</u>																	
Deprecation	Recommended Migration Action																
Support for the following tsx tags in the JavaServer Pages (JSP) engine are deprecated: <ul style="list-style-type: none"> repeat dbconnect dbquery getProperty userid passwd dbmodify 	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 Version 6.0, and the tag library is shipped with the product. Use the following table as a guideline for converting tsx tags to JSTL tags: <table border="0"> <tr> <td>tsx tag</td> <td>JSTL tag</td> </tr> <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; c:out value="\${book.title}" for example, where book is the current index in the result set</td> </tr> <tr> <td>tsx:userid</td> <td>Use the user attribute of the setDataSource tag</td> </tr> <tr> <td>tsx:passwd</td> <td>Use the password attribute of the setDataSource tag</td> </tr> <tr> <td>tsx:dbmodify</td> <td>sql:update</td> </tr> </table>	tsx tag	JSTL tag	tsx:repeat	c:forEach	tsx:dbconnect	sql:setDataSource	tsx:dbquery	sql:query	tsx:getProperty	Use standard EL syntax; c:out value="\${book.title}" for example, where book is the current index in the result set	tsx:userid	Use the user attribute of the setDataSource tag	tsx:passwd	Use the password 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; c:out value="\${book.title}" for example, where book is the current index in the result set																
tsx:userid	Use the user attribute of the setDataSource tag																
tsx:passwd	Use the password attribute of the setDataSource tag																
tsx:dbmodify	sql:update																
The following back-end IDs are deprecated: <ul style="list-style-type: none"> SQL92 (1992 SQL Standard) SQL99 (1999 SQL Standard) 	Use other back-end IDs.																
<u>Application Services Features</u>																	
Deprecation	Recommended Migration Action																
The JRAS Extensions API is deprecated. No further enhancements are planned for JRAS support.	Use the equivalent function in the java.util.logging package (JSR47).																
The UDDI Version 2 EJB interface to the UDDI Registry is deprecated.	There is no replacement for the EJB interface. This interface is included in WebSphere Application Server Version 6.0 for compatibility with Version 5.x. Users do not need to take any specific actions and can continue to use the Version 2 EJB API, but they should be aware that it does not include any UDDI functionality that is new to UDDI Version 3 and the interface might be removed in a future release of WebSphere Application Server.																
The UDDI4J Version 2 class library, the uddi4jv2.jar file, is deprecated.	Start using the Version 3 UDDI APIs. A client library is provided to simplify constructing and sending UDDI Version 3 requests from Java. This is the IBM UDDI Version 3 Client for Java, provided in uddiv3client.jar. The UDDI4J APIs can still be used, but you should be aware that they do not provide access to any of the new UDDI Version 3 functionality and they might be removed in a future release of WebSphere Application Server.																
All of the low-level UDDI Utility Tools (UUT) APIs, such as BusinessStub, ServiceStub, and so on, are deprecated. These are all replaced with the high-level PromoterAPI interface in the com.ibm.uddi.promoter package.	Start using the PromoterAPI interface in the com.ibm.uddi.promoter package 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.																

<p>The following methods in the J2EE Connector Architecture runtime are deprecated:</p> <ul style="list-style-type: none"> com.ibm.ws.management.descriptor.xml.ConnectionFactory.xml (getPoolContents and getAllPoolContents methods) com.ibm.websphere.j2c.ConnectionManager interface com.ibm.websphere.j2c.ConnectionEventListener interface <p>Also, container-managed authentication aliases on a J2C Connection Factory or datasource are deprecated.</p>	<p>The methods are replaced as follows:</p> <ul style="list-style-type: none"> getPoolContents and getAllPoolContents are replaced with showPoolContents and showAllPoolContents. ConnectionManager interface is replaced with J2EE Connector Architecture 1.5 LazyAssociatableConnectionManager interface. ConnectionEventListener interface is replaced with J2EE Connector Architecture 1.5 LazyEnlistableConnectionManager interface. <p>For container-managed authentication aliases, specify the container-managed credentials in the application's resource binding information.</p>
<p>The ApplicationProfile property on the WorkManager panel in the administrative console is deprecated.</p>	<p>See the information center for the differences between application profiling in Version 5.x and Version 6.0.x.</p>
<p>Two items from the DataSource panel in the administrative console are deprecated:</p> <ul style="list-style-type: none"> Container-Managed Authentication Alias DefaultPrincipleMapping 	<p>Define Container-Managed Authentication Alias and DefaultPrincipleMapping properties on the Resource Reference.</p>
<p>All classes in the com.ibm.websphere.servlet.filter package are deprecated:</p> <ul style="list-style-type: none"> ChainedRequest ChainedResponse ChainerServlet ServletChain 	<p>Rearchitect your 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 truncating responses.</p>
<p>MIME filtering is deprecated. MIME filters were first introduced in WebSphere Application Server Version 3.5 as a way for servlets to embellish, truncate, and modify the responses generated by other servlets, based on the MIME types of the output content.</p>	<p>javax.servlet.filters, which were introduced in the Servlet 2.3 specification, allow users to plug in filters that can intercept requests to and responses from servlets. They also have the capability to modify content flowing in either direction.</p> <p>javax.servlet.filters maintain all the functionality of MIME filters. javax.servlet.filters are standard APIs, and are supported by all compliant application servers.</p> <p>Refer to the Servlet 2.3 specification for more information.</p>
<p>Container-managed persistence (CMP) entity beans configured with method level access intent might run into data access problems, like deadlock. Therefore, the method level access intent is deprecated.</p>	<p>Reconfigure CMP entity beans to use bean level access intent, or reconfigure application profiles with WebSphere Application Server Tool (AST).</p>
<p>All the methods and fields in com.ibm.websphere.product.product and com.ibm.websphere.product.buildInfo classes are deprecated. Therefore, 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:</p> <ul style="list-style-type: none"> 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() 	<p>Use the following supported methods from com.ibm.websphere.product.WASDirectory:</p> <ul style="list-style-type: none"> public WASProductInfo getWASProductInfo(String id) public boolean isThisProductInstalled(String id) public WASProductInfo[] getWASProductInfoInstances() public String getWasLocation() <p>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:</p> <ul style="list-style-type: none"> com.ibm.websphere.product.WASDirectory.getName(String) com.ibm.websphere.product.WASDirectory.getVersion(String) com.ibm.websphere.product.WASDirectory.getBuildLevel(String) com.ibm.websphere.product.WASDirectory.getBuildDate(String)
<p>Data access beans, which are shipped with WebSphere Application Server in the databeans.jar file, are deprecated.</p>	<p>Instead of using data access beans, you should use Service Data Objects (SDO).</p>
<p>The reloadInterval and reloadingEnabled attributes of the IBM deployment descriptor extensions are deprecated, including both the WAR file extension (WEB-INF/ibm-web-ext.xmi) and the application extension (META-INF/ibm-application-ext.xmi).</p>	<p>Instead of using deployment descriptor extensions, you should use the reload enable and interval options provided during application deployment.</p>
<p>The com.ibm.websphere.servlet.session.UserTransactionWrapper API is deprecated.</p>	<p>There is no replacement for this API. The UserTransaction object can be placed directly into the HTTP session without using a wrapper.</p>
Security Features	
Deprecation	Recommended Migration Action
<p>SOAP-Security (XML digital signature) based on Apache SOAP implementation is deprecated.</p>	<p>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.</p>

Web Service Security (WSS) draft 13 specification-level support is deprecated in favor of the WSS 1.0 implementation.	<p>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.</p> <p>WSS 1.0 is only supported in J2EE 1.4 applications. Therefore, 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 Version 6.0; the migration has to be done manually.</p> <p>The following SPI has also been deprecated:</p> <p style="text-align: center;">com.ibm.wsspi.wssecurity.config.KeyLocator</p> <p>You need to migrate your implementation to the new SPI for WSS 1.0 support in Version 6.0:</p> <p style="text-align: center;">com.ibm.wsspi.wssecurity.keyinfo.KeyLocator</p> <p>Finally, the Java Authentication and Authorization Service (JAAS) LoginModule implementation needs to be migrated to the new programming model for JAAS LoginModule in Version 6.0.</p>
---	--

System Administration Features

Deprecation	Recommended Migration Action
Configuring resources under cell scope is deprecated.	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.	There is no replacement for the depl.extension.reg option. In Version 5.x, this option was a no-op. For the installdir option, use the installed.ear.destination option instead.

Performance Features

Deprecation	Recommended Migration Action
The PMI Client API, which was introduced in Version 4.0 to programmatically gather performance data from WebSphere Application Server, is deprecated.	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.

Features deprecated in Version 5.1.1

<u>Application Programming Model and Container Support Features</u>	
Deprecation	Recommended Migration Action
The Web services gateway customization API is deprecated.	Plan over time to replace your existing filters with a combination of JAX-RPC handlers and service integration bus mediations.
<u>Application Services Features</u>	
Deprecation	Recommended Migration Action
<p>The following Java Database Connectivity (JDBC) drivers are deprecated:</p> <ul style="list-style-type: none"> • MS SQL Server 2000 Driver for JDBC • SequeLink JDBC driver for MS SQL Server 	If you are using either of these JDBC drivers and still want to use MS SQL Server as your database, switch to the Connect JDBC driver. You can purchase the Connect JDBC driver from DataDirect Technologies; or you can use the Connect JDBC driver that is shipped with WebSphere Application Server, which is free for use with WebSphere Application Server.

Features deprecated in Version 5.1

<u>Installation and Migration Tools</u>	
Deprecation	Recommended Migration Action

The Application Assembly Tool (AAT) that is used for developing J2EE applications is replaced with the Assembly Tool (ATk) component of the Application Server Toolkit (ASTk).	Instead of running the Application Assembly Tool, users will install and run the Assembly Toolkit component of the Application Server Toolkit. ASTk is based on the Eclipse framework. Upon starting the ASTk, the J2EE function is found by opening the J2EE Perspective.
JDOM (a Java representation of an XML document that provides an API for efficient reading, manipulating, and writing documentation) is deprecated. The currently packaged version of JDOM in WebSphere Application Server will not be packaged in future releases.	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.
The C++ Object Request Broker (ORB), the C++ library for IDL value types and the WebSphere Application Server C++ security client are deprecated. 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. In addition to the preceding information, the CORBA C++ client feature will be removed from the Application Clients installation image in future releases.	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++ value-type library. Customers that require such functionality must provide or develop their own. 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: <ul style="list-style-type: none"> CORBA Interoperability Samples documentation IBM WebSphere Application Servers CORBA Interoperability white paper
IBM Cloudscape Version 5.1.x is deprecated.	Use the Cloudscape Network Server JDBC driver.
Servers and Clustering Features	
Deprecation	Recommended Migration Action
IBM HTTP Server (IHS) Version 1.3.x is deprecated.	If you are using IHS Version 1.3.x with modules that: <ul style="list-style-type: none"> are shipped as part of IHS Version 1.3.x packages, you do not need to take any action to migrate those modules. are supplied by a third party (including other IBM products), you need to obtain IHS/Apache 2 versions of these modules from the third party. have been customized or are inhouse, you need to port these modules to the new IHS/Apache 2 API.
Application Programming Model and Container Support Features	
Deprecation	Recommended Migration Action
Bean Scripting Framework (BSF) JavaServer Pages (JSP) execution and debugging functionality is deprecated.	The functionality will need to be rearchitected if you are using the JavaScript, Tcl, and Python languages. If using BSF scripting in your own custom applications, they will be unaffected. Custom scripts written for the WebSphere Application Server administrative console will also be unaffected. This functionality will continue to exist in WebSphere Application Server Version 5.1 and succeeding releases until Version 6.0. If debugging JSP files, you might have to restart the application 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)	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.

<p>Web container 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.</p> <p>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.</p> <ul style="list-style-type: none"> • Old method signature <pre>public String getStackTrace(); // returns a String representation of the exception stack</pre> • New method signature (JDK 1.4, WebSphere Application Server 5.1) <pre>public StackTraceElement[] getStackTrace(); // returns an array of stack trace elements</pre> • Replacement method (WebSphere Application Server 5.1) (a replacement method that carries on the old functionality has been provided): <pre>public String getStackTraceAsString(); // returns a String representation of the Exception Stack</pre> 	<p>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.</p>
Application Services Features	
Deprecation	Recommended Migration Action
<p>Data access binaries -- Common Connector Framework:</p> <p>The following JAR files are deprecated:</p> <ul style="list-style-type: none"> • ccf.jar • cc2.jar • recjava.jar • eablib.jar 	<p>The J2EE Connector Architecture solution should be used instead of the Common Connector Framework.</p>
<p>Setting the XA partner log directory using the TRANLOG_ROOT variable is deprecated.</p>	<p>The setting currently stored in the TRANLOG_ROOT variable (if any) will need to be added to the Transaction Service panel for all servers that need 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.</p>
Security Features	
Deprecation	Recommended Migration Action
<p>The API is deprecated for com.ibm.websphere.security.auth.WSPincipal.getCredential().</p>	<p>Instead of getting the WSCredential from the principal, you should now use one of the following methods to get the Subject that contains the WSCredential:</p> <ul style="list-style-type: none"> • 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 as follows: <pre>com.ibm.websphere.security.auth.WSSubject.getRunAsSubject()</pre> <p>and</p> <pre>com.ibm.websphere.security.auth.WSSubject.getCallerSubject()</pre> <p>respectively.</p>
<p>Security programming interface deprecations:</p> <ul style="list-style-type: none"> • The interface is deprecated in com.ibm.websphere.security.auth.WSSecurityContext. • The exception is deprecated in com.ibm.websphere.security.auth.WSSecurityContextException. • The class is deprecated in com.ibm.websphere.security.auth.WSSecurityContextResult. 	<p>Use Java Authentication and Authorization Service (JAAS) for all authentication related functionality.</p>

The Integrated Cryptographic Services Facility (ICSF) authentication mechanism is deprecated in Version 5.1.	Use the Lightweight Third Party Authentication (LTPA) mechanism.
System Administration Features	
Deprecation	Recommended Migration Action
The following class is deprecated: com.ibm.websphere.rsadapter.DB2390DataStoreHelper	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 when you are accessing data, you should now use the DB2UniversalDataStoreHelper class.

Features deprecated in Version 5.0.2

Application Programming Model and Container Support Features	
Deprecation	Recommended Migration Action
Apache SOAP channel in Web services gateway.	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: <ul style="list-style-type: none"> • soap.jar • wsoap.jar 	See the information center for more information.
Application Services Features	
Deprecation	Recommended Migration Action
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 <pre>public int processSQL(java.lang.String sqlString, int isolateLevel, boolean addForUpdate, boolean addextendedForUpdateSyntax); public DataStoreAdapterException mapException(DataStoreAdapterException e);</pre> com.ibm.websphere.rsadapter.GenericDataStoreHelper <pre>public int processSQL(java.lang.String sqlString, int isolateLevel, boolean addForUpdate, boolean addextendedForUpdateSyntax); public DataStoreAdapterException mapException(DataStoreAdapterException e);</pre> com.ibm.websphere.rsadapter.WSCallHelper <pre>public static DataStoreHelper createDataStoreHelper(String dsClassName)</pre>	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, use the getDataStoreHelper(datasource) method to get a DataStoreHelper object.
System Administration Features	
Deprecation	Recommended Migration Action
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.	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.	There is no replacement for this command.

Features deprecated in Version 5.0.1

<u>Application Services Features</u>	
Deprecation	Recommended Migration Action
<p>Data access programming interfaces in com.ibm.websphere.rsadapter.</p> <p>Relational resource adapter interface (com.ibm.websphere.rsadapter).</p> <p>Methods have been deprecated in these types:</p> <pre>com.ibm.websphere.rsadapter.DataStoreHelper public int processSQL(java.lang.String sqlString, int isoLevel);</pre> <pre>com.ibm.websphere.rsadapter.GenericDataStoreHelper public int processSQL(java.lang.String sqlString, int isoLevel);</pre> <pre>com.ibm.websphere.rsadapter.DB2390DataStoreHelper public int processSQL(java.lang.String sqlString, int isoLevel);</pre>	<p>These relational resource adapter deprecated methods do not impact the application.</p> <p>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.</p>

Features deprecated in Version 5.0

<u>Application Services Features</u>	
Deprecation	Recommended Migration Action
<p>The following three methods from com.ibm.websphere.appprofile.accessintent.AccessIntent are deprecated:</p> <pre>public boolean getPessimisticUpdateHintWeakestLockAtLoad(); public boolean getPessimisticUpdateHintNoCollision(); public boolean getPessimisticUpdateHintExclusive();</pre> <p>This is a base API.</p>	<p>Rather than using the three deprecated methods on the AccessIntent interface, developers should use the following method from the same interface:</p> <pre>public int getPessimisticUpdateLockHint();</pre> <p>The possible return values are defined on the AccessIntent interface:</p> <pre>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;</pre>
<p>Web application programming interfaces -- Various Version 5.x methods in com.ibm.websphere.ServletErrorReport</p>	
<u>Security Features</u>	
Deprecation	Recommended Migration Action
<p>The com.ibm.websphere.security.CustomRegistry interface is deprecated.</p>	<p>Use the com.ibm.websphere.security.UserRegistry interface.</p> <p>See "Migrating custom user registries" on page 59.</p>
<u>Performance Features</u>	
Deprecation	Recommended Migration Action
<p>Performance Monitoring Infrastructure -- Various Version 5.x public methods in:</p> <ul style="list-style-type: none"> • com.ibm.websphere.pmi.stat.StatsUtil • com.ibm.websphere.pmi.PmiJmxTest • com.ibm.websphere.pmi.client.PmiClient 	<p>These methods are replaced as follows:</p> <ul style="list-style-type: none"> • com.ibm.websphere.pmi.stat.StatsUtil There is no replacement for StatsUtil. • com.ibm.websphere.pmi.PmiJmxTest Use PmiClient.findConfig() instead. • com.ibm.websphere.pmi.client.PmiClient The getNLSValue (String key) is replaced with getNLSValue (String key, String moduleID).

Removal list

The following tables describe what is removed—such as features, APIs, scripting interfaces, tools, wizards, publicly exposed configuration data, naming identifiers, and constants. Where possible, the recommended replacement is identified.

- "Features removed in Version 6.1" on page 12

- “Features removed in Version 6.0” on page 14

Features removed in Version 6.1

Feature	Recommended Migration Action
com.ibm.websphere.security.CustomRegistry interface	Use the com.ibm.websphere.security.UserRegistry interface. See “Migrating custom user registries” on page 59.
Support for the z/OS Secure Authentication Service (z/SAS) IIOF security protocol	Use the Common Secure Interoperability Version 2 (CSlv2) protocols.
Support for the Common Connector Framework (CCF)	Use the J2EE Connector Architecture (JCA) solution.
Support for the IBM Cloudscape Version 5.1.x database	Use the IBM Cloudscape Version 10.1 database. This database provides Derby Version 10.1 binaries, NLS enablement, QA, and IBM problem support. Note: When using WebSphere Application Server, you will see the term “Derby” rather than “Cloudscape” used in places such as the administrative console, the ejbdeploy command, and others.
Log Analyzer, the tool that was previously provided for viewing and analyzing the activity or service log file	Use the Log and Trace Analyzer tool for Eclipse in the Application Server Toolkit (AST). This tool is installable from the AST launchpad console. For more information, go to Application Server Toolkit > Detecting and analyzing runtime problems > Log and Trace Analyzer in this information center.
Mozilla Rhino JavaScript (js.jar)	Use the Rhino code available from Mozilla. Go to Rhino: JavaScript for Java and get the latest copy of Rhino.
Java Document Object Model (JDOM)	Use the code available from the JDOM organization. Go to JDOM, get the latest copy of JDOM, and bundle it inside your application.
DB2 for zOS Local JDBC Provider (RRS)	Use the DB2 Universal JDBC Driver Provider. See “Premigration considerations” on page 81 for more information. Also see “Migrating from the JDBC/SQLJ Driver for OS/390 and z/OS to the DB2 Universal JDBC Driver” in the Information Management Software for z/OS Solutions Information Center.
Class preloading function	No migration action is necessary.
The following samples from the Samples Gallery: <ul style="list-style-type: none"> • Adventure Builder • Greenhouse by WebSphere • WebSphere Bank The following technology samples from the Samples Gallery: <ul style="list-style-type: none"> • Bean-Managed Persistence (BMP) • Container-Managed Persistence (CMP) 1.1 • Container-Managed Persistence (CMP) 2.1 • Container-Managed Relationships (CMR) • EJB Time • Filter Servlet • JavaServer Pages (JSP) 2.0 • Message-Driven Beans (MDB) • Pagelist Servlet • Simple JavaServer Pages (JSP) • Simple Servlet • Stateful Session • TagLib 	No migration action is necessary.

Feature	Recommended Migration Action
<p>The following Multiple Virtual Storage (MVS) console display commands related to existing sessions:</p> <ul style="list-style-type: none"> • DISPLAY,SESSIONS • DISPLAY,SESSIONS,LISTENERS • DISPLAY,SESSIONS,SERVER • DISPLAY,SESSIONS,SERVER,TCPIIOP • DISPLAY,SESSIONS,SERVER,TCPIIOP,LIST • DISPLAY,SESSIONS,SERVER,LOCALIOP • DISPLAY,SESSIONS,SERVER,LOCALIOP,LIST • DISPLAY,SESSIONS,SERVER,SSLIOP • DISPLAY,SESSIONS,SERVER,SSLIOP,LIST • DISPLAY,SESSIONS,SERVER,HTTP • DISPLAY,SESSIONS,SERVER,HTTP,LIST • DISPLAY,SESSIONS,SERVER,HTTPS • DISPLAY,SESSIONS,SERVER,HTTPS,LIST 	<p>Modify all automation or other processing that uses these commands to use the following new set of commands:</p> <ul style="list-style-type: none"> • DISPLAY,LISTENERS • DISPLAY,CONNECTIONS • DISPLAY,CONNECTIONS,NAME='name' • DISPLAY,CONNECTIONS,LIST • DISPLAY,CONNECTIONS,LIST,NAME='name'
<p>The following configuration variables:</p> <ul style="list-style-type: none"> • com_ibm_userRegistries_type • com_ibm_userRegistries_LDAPUserRegistry_realm • com_ibm_userRegistries_CustomUserRegistry_realm • control_region_ssl_thread_pool_size • control_region_security_enable_trusted_applications • nonauthenticated_clients_allowed • security_zSAS_ssl_repertoire • security_sslType1 • security_sslClientCerts_allowed • security_kerberos_allowed • security_userid_password_allowed • security_userid_passticket_allowed • security_assertedID_IBM_accepted • security_assertedID_IBM_sent • protocol_http_max_keep_alive_connections • protocol_http_max_connect_backlog • protocol_https_transport_class_mapping_file • protocol_https_max_keep_alive_connections • protocol_https_max_connect_backlog • protocol_iiop_no_local_copies 	<p>No migration action is necessary.</p>

Features removed in Version 6.0

Component	Classes and interfaces
activity	com.ibm.ws.activity.ActivityConstants com.ibm.ws.activity.ActivityService com.ibm.ws.activity.ActivityServiceInitializer com.ibm.ws.activity.ActivityTrace com.ibm.ws.activity.GlobaldImpl com.ibm.ws.activity.HighlyAvailableServiceManager com.ibm.ws.activity.HLSLiteDataInterface com.ibm.ws.activity.HLSLiteExtended com.ibm.ws.activity.HLSLiteInfo com.ibm.ws.activity.j2ee_activity_specific_data com.ibm.ws.activity.j2ee_activity_specific_dataHelper com.ibm.ws.activity.ServiceMigration com.ibm.ws.activity.VUTrace com.ibm.ws.activity.WebSphereServiceManager com.ibm.ws.activity.WebSphereUserActivity com.ibm.ws.javax.activity.ActionErrorException com.ibm.ws.javax.activity.ActionNotFoundException com.ibm.ws.javax.activity.ActivityCoordinator com.ibm.ws.javax.activity.ActivityInformation com.ibm.ws.javax.activity.ActivityManager com.ibm.ws.javax.activity.ActivityNotProcessedException com.ibm.ws.javax.activity.ActivityPendingException com.ibm.ws.javax.activity.ActivityToken com.ibm.ws.javax.activity.CompletionStatus com.ibm.ws.javax.activity.ContextPendingException com.ibm.ws.javax.activity.CoordinationInformation com.ibm.ws.javax.activity.Globald com.ibm.ws.javax.activity.InvalidParentContextException com.ibm.ws.javax.activity.InvalidStateException com.ibm.ws.javax.activity.NoActivityException com.ibm.ws.javax.activity.NoImplementException com.ibm.ws.javax.activity.NotOriginatorException com.ibm.ws.javax.activity.Outcome com.ibm.ws.javax.activity.PersistentActivityCoordinator com.ibm.ws.javax.activity.PropertyGroupContext com.ibm.ws.javax.activity.PropertyGroupRegisteredException com.ibm.ws.javax.activity.PropertyGroupUnknownException com.ibm.ws.javax.activity.ServiceAlreadyRegisteredException com.ibm.ws.javax.activity.ServiceInformation com.ibm.ws.javax.activity.ServiceNotRegisteredException com.ibm.ws.javax.activity.Signal com.ibm.ws.javax.activity.SignalSetActiveException com.ibm.ws.javax.activity.SignalSetInactiveException com.ibm.ws.javax.activity.SignalSetUnknownException com.ibm.ws.javax.activity.Status com.ibm.ws.javax.activity.SystemException com.ibm.ws.javax.activity.TimeoutRangeException com.ibm.ws.javax.activity.UserActivity com.ibm.ws.javax.activity.coordination.Action com.ibm.ws.javax.activity.coordination.RecoverableAction com.ibm.ws.javax.activity.coordination.ServiceManager com.ibm.ws.javax.activity.coordination.SignalSet com.ibm.ws.javax.activity.coordination.SubordinateSignalSet com.ibm.ws.javax.activity.propertygroup.PropertyGroup com.ibm.ws.javax.activity.propertygroup.PropertyGroupManager com.ibm.ws.javax.ejb.ActivityCompletedLocalException com.ibm.ws.javax.ejb.ActivityRequiredLocalException com.ibm.ws.javax.ejb.InvalidActivityLocalException
admin	com.ibm.websphere.management.application.EarUtils
als	com.ibm.websphere.als.BufferManager
anttasks	com.ibm.websphere.ant.tasks.endptEnabler.Property com.ibm.websphere.ant.tasks.Java2WSDL.Mapping com.ibm.websphere.ant.tasks.Messages com.ibm.websphere.ant.tasks.WSDL2Java.Mapping
dynacache	com.ibm.websphere.servlet.cache.CacheConfig

Component	Classes and interfaces
ras	com.ibm.ras.RASConsoleHandler com.ibm.ras.RASEnhancedMessageFormatter com.ibm.ras.RASEnhancedTraceFormatter com.ibm.ras.RASErrorHandler com.ibm.ras.RASFileHandler com.ibm.ras.RASFormatter com.ibm.ras.RASHandler com.ibm.ras.RASMessageFormatter com.ibm.ras.RASMultiFileHandler com.ibm.ras.RASSerialFileHandler com.ibm.ras.RASSocketHandler com.ibm.ras.RASTextAreaHandler com.ibm.ras.RASTraceFormatter com.ibm.websphere.ras.WsOrbRasManager
security	com.ibm.websphere.security.AuthorizationTable com.ibm.websphere.security.FileRegistrySample com.ibm.websphere.security.SecurityProviderException com.ibm.websphere.security.WASPrincipal com.ibm.websphere.security.auth.AuthDataFileEnc
userprofile	com.ibm.websphere.userprofile.UserProfile com.ibm.websphere.userprofile.UserProfileCreateException com.ibm.websphere.userprofile.UserProfileExtender com.ibm.websphere.userprofile.UserProfileFinderException com.ibm.websphere.userprofile.UserProfileManager com.ibm.websphere.userprofile.UserProfileProperties com.ibm.websphere.userprofile.UserProfileRemoveException
Scheduler API	com.ibm.websphere.scheduler.pmi.SchedulerPmiModule com.ibm.websphere.scheduler.pmi.SchedulerPerf
Scheduler API	com.ibm.websphere.scheduler.MessageTaskInfo.setJMSPriority()
ObjectPool APIs	com/ibm/websphere/objectpool/pmi/ObjectPoolPerf.java com/ibm/websphere/objectpool/pmi/ObjectPoolPmiModule.java
Asynchronous Beans APIs	com/ibm/websphere/asynchbeans/pmi/AlarmManagerPerf.java com/ibm/websphere/asynchbeans/pmi/AsynchBeanPerf.java com/ibm/websphere/asynchbeans/pmi/SubsystemMonitorManagerPerf.java com/ibm/websphere/asynchbeans/pmi/SubsystemMonitorPerf.java com/ibm/websphere/asynchbeans/pmi/AlarmManagerPmiModule.java com/ibm/websphere/asynchbeans/pmi/AsynchBeanPmiModule.java com/ibm/websphere/asynchbeans/pmi/SubsystemMonitorManagerPmiModule.java com/ibm/websphere/asynchbeans/pmi/SubsystemMonitorPmiModule.java

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

Use the documentation provided to answer your questions about migration.

Hold your cursor over the task icon () to see a description of the task. The task preview feature is unavailable for Mozilla Web browsers.

Determine what configuration information needs to change	Documentation
Migrate Web server configurations	Documentation
Identify deprecated features that you might be using	Documentation
Review the software and hardware prerequisites	Documentation
Review supported coexistence scenarios	Documentation
Obtain valid port settings for coexistence	Documentation
Interoperate across product versions	Documentation

Chapter 3. Migrating applications

Use this section to learn about migrating applications.

Web applications

Migrating to Java 2 Standard Edition (J2SE) 5

This product version supports the Java 2 Standard Edition (J2SE) 5 specification. Its Java virtual machine provides a Java language compiler and execution environment. Decide whether your new and existing applications will take advantage of the capabilities added by J2SE 5, adjust the JIT mode if necessary, and begin the transition from deprecated functions.

For an introduction to J2SE 5, see the "J2SE 5 in a Nutshell" article on the Sun site at <http://java.sun.com/developer/technicalArticles/releases/j2se15/>. The following JSRs are new in J2SE 5:

- JSR 003: The JMX 1.2 specification. Packages: javax.management.*
- JSR 013: Additions to java.math for improved arithmetic operations using BigDecimal
- JSR 028: The Java SASL packages: javax.security.sasl
- JSR 114: JDBC Rowset implementations that specify rowset more completely
- JSR 133: Java Memory Model and Thread Specification Revision
- JSR 160: JMX remote API, V1.0
- JSR 163: Java Platform Profiling Architecture, JVMTI (replaces JVMPi)
- JSR 166: Concurrency utilities. Packages: java.util.concurrent.*
- JSR 174: Monitoring and Management Specification for the Java Virtual Machine
- JSR 175: A Metadata Facility for the Java Programming Language
- JSR 200: Network Transfer Format for Java Archives
- JSR 201: Extending the Java Programming Language with Enumerations, Autoboxing, Enhanced for Loops and Static Import
- JSR 206: Java API for XML Processing (JAXP) 1.3
- JSR 204: Unicode Supplementary Character Support

The new virtual machine specification adds several features and functions to benefit application developers, such as generics, auto-boxing of primitives, annotations (API documentation metadata for code generation), and support for enumerated types. This makes development quicker, easier, and less error prone. For example, generics should help eliminate issues with ClassCastException from items like vectors, as generics based containers will allow compile-time catching of incorrect assignment or casting. (For developers familiar with the C++ language, generics are a new Java language function similar to C++ templates.)

For details, see the J2SE 5 application programming interface documentation on the Sun site at <http://java.sun.com/j2se/1.5.0/docs/api/index.html>. See also <http://java.sun.com/j2se/1.5.0/docs/index.html>.

- Decide whether to take advantage of new J2SE 5 capabilities in your applications.

Applications using the new language features and J2SE 5 can be deployed only to Version 6.1 nodes, as earlier product versions do not provide the J2SE 5 virtual machine.

The J2EE 1.4 specification does not take into account the new language features. Therefore, the usage of generics based types should not be used with public EJB interfaces that are exposed on the home, stubs, and so forth.

If the code being developed must run on multiple J2SE levels, use only the API specification for the minimum J2SE level, such as J2SE 1.4, to avoid inadvertent usage of classes and methods that are not part of all of the required J2SE levels. Failure to do so may cause application breakage on older J2SE implementations.

Applications that access classes and APIs internal to the Java virtual machine could have problems. These classes and APIs are not covered by the J2SE 5 specification and are therefore subject to change. Packages with prefixes such as 'com.sun.*' are considered internal. Additionally, direct use of implementations of XML and XSL parsers is strongly discouraged, such as direct use of Xerces and Xalan classes that provide the JAXP implementation for the virtual machine. The direct parser APIs also are considered internal and subject to change. Applications should rely only on the JAXP APIs defined in the J2SE 1.4 and J2SE 5 API documentation. If your application requires a specific version of Xerces or Xalan, or some other XML/XSL parser package, then embed the parser within your application's WEB-INF/lib directory and set the appropriate class loading mode in your application deployment so that for your application the XML parser APIs are loaded from the application class path, not the Java virtual machine bootstrap class path. Failure to follow this guideline can cause significant problems when trying to migrate to a new J2SE level.

- Compile J2SE 5 applications to run on older Java virtual machine levels by setting the compiler modes.

When compiling applications that are built with J2SE 5 that are intended for running on older J2SE specifications, be sure to specify '-source' and '-target' modes for the J2SE 5 compiler. Doing so ensures that the bytecode generated is compatible with the earlier Java virtual machine.

For example, if the target Java virtual machine is at 1.4.2 level, when you compile applications with J2SE 5, you should specify '-source 1.4', and 'target 1.4' to generate bytecode compatible with 1.4.2. This does not handle the usage of packages, classes, or functions new to J2SE 5. It only addresses bytecode output. Developers must take care in what APIs they are using from the J2SE packages if they intend to run the application on multiple Java virtual machine specification levels.

- Address incompatibilities in previously compiled J2SE 1.4 based applications.

Java (TM) 2 Technology Edition, Version 5.0 is upwards binary-compatible with Java (TM) 2 Technology Edition, Version 1.4.2, except for the incompatibilities documented by Sun Microsystems at <http://java.sun.com/j2se/1.5.0/compatibility.html#binary>. Most of the incompatibilities refer to compiling classes at a JDK 5 level using a target of 1.5. "Almost all existing programs should run on J2SE 5.0 without modification," to cite the Sun documentation. As a migration reference, the J2SE 1.4 and J2SE 5 application programming interface documentation is available on the Sun site at <http://java.sun.com/j2se/1.4.2/docs/api/index.html> and <http://java.sun.com/j2se/1.5.2/docs/api/index.html>, respectively.

Here are the most notable source compatibility problems.

- **Variables named 'enum.'** The word 'enum' has become a language keyword. It now will cause a compiler fault if used as a variable name. Consider specifying '-source 1.4' for compilations with J2SE 5 until you can correct the variable names. While using -source 1.4, all new JDK 5 language constructs are disabled and cannot be used in the source code.
- **Ambiguous references to classes with base names of 'Proxy,' 'Queue,' or 'Formatter.'** You might encounter compile-time errors if you import java.net.* and then use other classes with a base name of Proxy without fully qualifying the latter class names. The errors are because java.net.* now contains java.net.Proxy.

In similar circumstances, you might encounter errors importing java.lang.reflect.* Note also that a new java.util.Queue class in J2SE 5 conflicts with other Queue package names, such as javax.jms.Queue.

You might encounter compile-time errors if you import java.util.* and then use other classes with a base name of Formatter, without fully qualifying the latter class names. The errors occur because java.util.* now contains java.util.Formatter, a class added in the J2SE 5 spec.

- Start the transition from deprecated JVMDI and JVMPI functions to JVMTI.

J2SE 5 deprecates some functions that were available for public usage in previous J2SE specifications. JVMDI and JVMPI are deprecated in J2SE 5 and might be removed in the next major release of the J2SE specification. Any new development and tool sets should begin moving to JVMTI. Migrate any of

your native (JNI) performance profiling libraries to the new JVMTI API described at <http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/index.html>.

- Update your use of the Java command line interface.

The command-line interfaces for the J2SE 5 level have not changed extensively from J2SE 1.4, although they vary among virtual machine vendors. You can find them in the `JAVA_HOME/bin` directory. Here are some notable command line options that are standard to all J2SE 5 implementations.

- For JVMTI, use `-agentlib` to load a native agent library that you specify.
- For JVMTI, use `-agentpath` to load the native agent library by the full path name
- For JVMTI, use `-javaagent` to load the Java programming language agent (see `java.lang.instrument` for details)
- See `apt -help` for information about this new command line supporting the annotations capability.
- See `javac -help` for information and updates to that command line.

- Update ANT tasks.

If you have created ANT tasks based on the `idtojava` ANT task shipped with prior versions of this product, you will need to ensure that it passes the proper parameters for J2SE 5 as it does for J2SE 1.4, to ensure the stubs/ties and skeletons it generates are compatible to earlier product releases.

Migrating servers from multi-broker replication domains to data replication domains

Use this task to migrate multi-broker replication domains to data replication domains. Any multi-broker domains that exist in your WebSphere Application Server environment were created with a previous version of WebSphere Application Server.

For HTTP session affinity to continue working correctly when migrating V5.x application servers to V6.1 application servers, you must upgrade all of the Web server plug-ins for WebSphere Application Server to the latest version before upgrading the application servers that perform replication.

After you upgrade your deployment manager to the latest version of WebSphere Application Server, you can create data replication domains only. Any multi-broker domains that you created with a previous release of WebSphere Application Server are still functional, however, you cannot create new multi-broker domains or replicators with the administrative console.

The different versions of application servers cannot communicate with each other. When migrating your servers to the current version of WebSphere Application Server, keep at least two application servers running on the previous version so that replication remains functional.

Perform this task on any multi-broker domains in your configuration after all of your servers that are using this multi-broker domain have been migrated to the current version of WebSphere Application Server. For more information about the differences between multi-broker domains and the data replication domains, see “Comparison of multi-broker versus data replication domains” on page 23.

The following examples illustrate the migration process for common configurations:

Migrating an application server configuration that uses an instance of data replication service in peer-to-peer mode

Use this migration path to migrate a replication domain that uses the default peer-to-peer configuration. Dynamic cache replication domains use the peer-to-peer topology.

Before you begin, migrate all the Web server plug-ins for your application server cluster to the current version.

1. Migrate one or more of your existing servers to the current version of WebSphere Application Server.

2. In the administrative console, create an empty data replication domain. Click **Environment > Replication domains > New** in the administrative console.
3. Add your migrated application servers to the new data replication domain. For example, if you are migrating 4 servers, migrate 2 servers first and add them to the new replication domain. Configure the servers to use the new domain by configuring the consumers of the replication domain.
4. When the new data replication domains are successfully sharing data, migrate the rest of the servers that are using the multi-broker replication domain to data replication domains.
5. Delete the empty multi-broker replication domain.

Migrating an application server configuration that uses an instance of the data replication service in client/server mode

Use this set of steps to migrate a replication domain that uses client/server mode.

Before you begin migrating a client/server mode replication domain, consider if migrating your replication domains might cause a single point of failure. Because you migrate the servers to the new type of replication domain one at a time, you risk a single point of failure if there are 3 or fewer application servers. Before migrating, configure at least 4 servers that use multi-broker replication domains. Perform the following steps to migrate the multi-broker domains to data replication domains:

1. Migrate one or more of your existing servers to the current version of WebSphere Application Server.
2. In the administrative console, click **Environment > Replication domains > New** to create an empty data replication domain.
3. Add your migrated servers to the new data replication domain. For example, if you are migrating 4 servers, migrate two of these servers and then add them to the new replication domain. Configure the servers to use the new domain by configuring the consumers of the replication domain.
4. Add a part of the clients to the new data replication domain.
5. When the new data replication domains are successfully sharing data, migrate the rest of the clients and servers that are using the multi-broker replication domain to data replication domains.
6. Delete the empty multi-broker replication domain.

Migrating a replication domain that uses HTTP session memory-to-memory replication that is overloaded at the application or web module level

1. Upgrade your deployment manager to the current version of WebSphere Application Server. All the application servers remain configured with the old multi-broker domains on the previous version of WebSphere Application Server.
2. In the administrative console, create an empty data replication domain. Click **Environment > Replication domains > New** in the administrative console.
3. Migrate each application server to the current version of WebSphere Application Server, one at a time. The remaining servers on the previous version of WebSphere Application Server can still communicate with each other, but not with the migrated servers. The migrated servers can also communicate with each other.
4. Continue migrating all of the servers to the current version of WebSphere Application Server. All of the application servers are still using multi-broker replication domains, so the features of data replication domains cannot be used.
5. Configure all of the application servers to use the new data replication domain, adding the application servers to the empty replication domain that you created.
6. Restart all of the application servers in the cluster.
7. Delete the empty multi-broker replication domain.

During this process, you might lose existing sessions. However, the application remains active through the entire process, so users do not experience down time during the migration. Create a new replication domain for each type of consumer. For example, create one replication domain for session manager and another replication domain for dynamic cache.

Comparison of multi-broker versus data replication domains

Data replication domains replace multi-broker domains for data replication between application servers in a cluster.

transition:

Any replication domains that are created with a previous version of WebSphere Application Server might be multi-broker domains. Migrate any multi-broker domains to the new data replication domains. Although you can configure existing multi-broker domains with the current version of WebSphere Application Server, after you upgrade your deployment manager, you can create only data replication domains in the administrative console.

Multi-broker and data replication domains both perform the same function, which is to replicate data across the consumers in a replication domain. Configure all the instances of replication that need to communicate in the same replication domain. You can also configure the session manager with both types of replication domains to use topologies such as peer-to-peer and client/server to isolate the function of creating and storing replicas on separate application servers. You can control the redundancy of replication for each type of replication domain. With a data replication domain, you can specify a specific number of replicas.

If you used multi-broker domains with earlier releases of WebSphere Application Server, use the following comparison chart to learn the differences between how V5.x and V6.x application servers use the two types of replication domains:

	V5.x application servers using replication domains	V6.x application servers using replication domains
Replication domain types	Uses only multi-broker replication domains for replication.	Servers that are using the current version of WebSphere Application Server can be configured to use both multi-broker replication domains and data replication domains for replication. The two types of domains provide backward compatibility with multi-broker domains that were created with a V5.x server. You should migrate any multi-broker domains to data replication domains.

	V5.x application servers using replication domains	V6.x application servers using replication domains
Data transport method	Uses multi-broker domain objects that contain configuration information for the internal Java Message Service (JMS) provider, which uses JMS brokers as replicators.	Uses data replication domain objects that contain configuration information to configure the high availability framework on WebSphere Application Server. The transport is no longer based on the JMS API. Therefore, no replicators and no JMS brokers exist. You do not have to perform the complex task of configuring local, remote, and alternate replicators. The earlier version of WebSphere Application Server did not support data replication domains. The current version of WebSphere Application Server can be configured to perform replication using old multi-broker domains by ignoring any JMS-specific configuration and by using the other parameters to configure replication through the high availability framework.
Replication domain configuration	The earlier version of WebSphere Application Server encourages the sharing of replication domains between different consumers, such as session manager and dynamic cache.	The current version of WebSphere Application Server encourages creating a separate replication domain for each consumer. For example, create one replication domain for session manager and another replication domain for dynamic cache. The only situation where you should configure one replication domain is when configuring session manager replication and stateful session bean failover. Using one replication domain in this case ensures that the backup state information of HTTP sessions and stateful session beans are on the same application servers.
Partial partitioning	You can configure partial partitioning. Partition the replication domain to filter the number of processes to send data.	Partial partitioning is deprecated. When using data replication domains, you can specify a specific number of replicas for each entry. However, if you specify a number of replicas larger than the number of backup application servers that are running, the number of replicas is the number of application servers that are running. After the number of application servers increases above your configured number of replicas, the number of replicas that are created is equal to the number that you specified.

	V5.x application servers using replication domains	V6.x application servers using replication domains
Domain sharing	Multiple data replication service (DRS) instances share multi-broker domains. A limitation exists on the number of multi-broker domains that you can create because every multi-broker domain contains at least one replicator. A maximum of one replicator can be on each application server.	All DRS instances in a replication domain use the same mode. Each replication domain must contain either client only and server only instances, or client and server instances only. For example, if one instance is configured to client and server, all other instances must be client and server. If one instance in a replication domain is configured to be a client only, you can add client only and server only instances, but not a client and server instance.

To migrate multi-broker domains to data replication domains, see the *Migrating, coexisting, and interoperating* PDF.

JavaServer Pages migration best practices and considerations

The standard JavaServer Pages (JSP) tags from JSP 1.1 such as `jsp:include`, `jsp:useBean`, and `<%@page %>`, will migrate successfully to JSP 2.0. However, there are several areas that must be considered when migrating JavaServer Pages. This topic discusses the areas that you must consider when migrating JavaServer Pages.

Classes from the unnamed or default package

As of JSP 2.0, referring to any classes from the unnamed or default package is not allowed. This can result in a translation error on some containers, specifically those that run in a JDK 1.4 or greater environment which will also break compatibility with some older JSP applications. However, as of JDK 1.4, importing classes from the unnamed package is not valid. See Java 2 Platform, Standard Edition Version 1.4.2 Compatibility with Previous Releases for details. Therefore, for forwards compatibility, applications must not rely on the unnamed package. This restriction also applies for all other cases where classes are referenced, such as when specifying the class name for a tag in a Tag Library Descriptor (TLD) file.

Page encoding for JSP documents

There have been noticeable differences in internationalization behavior on some containers as a result of ambiguity in the JSP 1.2 specification. However, steps were taken to minimize the impact on backwards compatibility and overall, the internationalization abilities of JSP files have been greatly improved.

In JSP specification versions prior to JSP 2.0, JSP pages in XML syntax, JSP documents, and those in standard syntax determined their page encoding in the same fashion, by examining the `pageEncoding` or `contentType` attributes of their page directive, defaulting to ISO-8859-1 if neither was present.

As of JSP 2.0, the page encoding for JSP documents is determined as described in section 4.3.3 and appendix F.1 of the XML specification, and the `pageEncoding` attribute of those pages is only checked to make sure it is consistent with the page encoding determined as per the XML specification. As a result of this change, JSP documents that rely on their page encoding to be determined from their `pageEncoding` attribute are no longer decoded correctly. These JSP documents must be changed to include an appropriate XML encoding declaration.

Additionally, in JSP 1.2, page encodings are determined on translation unit basis whereas in JSP 2.0, page encodings are determined on the basis of each file. Therefore, if the `a.jsp` file statically includes the `b.jsp` file, and a page encoding is specified in the `a.jsp` file but not in the `b.jsp` file, in JSP 1.2 the encoding

for the a.jsp file is used for the b.jsp file, but in JSP 2.0, the default encoding is used for the b.jsp file.

web.xml file version

The JSP container uses the version of the web.xml file to determine whether you are running a JSP 1.2 application or a JSP 2.0 application. Various features can behave differently depending on the version of the web.xml file. The following is a list of things JSP developers should be aware of when upgrading their web.xml file from version Servlet 2.3 to version Servlet 2.4:

1. EL expressions are ignored by default in JSP 1.2 applications. When you upgrade a Web application to JSP 2.0, EL expressions are interpreted by default. You can use the escape sequence `\$` to escape EL expressions that should not be interpreted by the container. Alternatively, you can use the `isELIgnored` page directive attribute, or the `<el-ignored>` configuration element to deactivate EL for entire translation units. Users of JSTL 1.0 must upgrade their taglib imports to the JSTL 1.1 uris or use the `_rt` versions of the tags, for example, use `c_rt` instead of `c` or `fmt_rt` instead of `fmt`.
2. Web applications that contain files with an extension of `.jspx` will have those files interpreted as JSP documents, by default. You can use the JSP configuration element `<is-xml>` to treat `.jspx` files as regular JSP pages, but there is no way to disassociate `.jspx` from the JSP container.
3. The escape sequence `\$` was not reserved in JSP 1.2. The output for any template text or attribute value that appeared as `\$` in JSP 1.2 was `\$`, however, the output now is just `$`.

jsp:useBean tag

WebSphere Application Server version 5.1 and later enforces more strict adherence to the specification for the `jsp:useBean` tag: with `type` and `class` attributes. Specifically, you should use the `type` attribute should be used to specify a Java type that cannot be instantiated as a `JavaBean`. For example, a Java type that is an abstract class, interface, or a class with no public no-args constructor. If the `class` attribute is used for a Java type that cannot be instantiated as a `JavaBean`, the WebSphere Application Server JSP container produces a unrecoverable translation error at translation time.

Generated packages for JSP classes

Any reliance on generated packages for JSP classes will result in non-portable JSP files. Packages for generated classes are implementation-specific and therefore you should not rely on these packages.

JspServlet class

Any reliance on the existence of a `JspServlet` class will cause unrecoverable error problems. WebSphere Application Server version 6.0 and later no longer uses a `JspServlet` class.

JavaServer Faces

JavaServer Faces (JSF) is a user interface framework or API that eases the development of Java based Web applications. WebSphere Application Server version 6.1 supports JavaServer Faces 1.1 at a runtime level, therefore using JSF reduces the size of the Web application since runtime binaries no longer need to be included in your Web application.

The JSF runtime also :

- Makes it easy to construct a user interface from a set of reusable user interface components
- Simplifies migration of application data to and from the user interface
- Helps manage user interface state across server requests
- Provides a simple model for wiring client-generated events to server-side application code
- Allows custom user interface components to be easily build and reused

The Sun JSF Reference Implementation provides the foundation of the code used for the JSF support in WebSphere Application Server. However, some dependencies on Jakarta APIs have been removed and replaced with Application Server specific solutions as a result of potential problems that may occur when Open Source APIs are included in the Application Server runtime. For example, when included in the Application Server runtime, these Open Source APIs are made available to all applications installed within the Application Server, therefore bringing versioning, support and legal issues. The version of the JSF runtime provided by the Application Server resides in the normal runtime library location and is available to all Web applications that leverage JSF APIs. The loading of the JSF servlet works in the same manner as if the runtime was packaged with the Web application.

The following open source dependencies are replaced with other APIs or in-house versions:

- Jakarta Commons BeanUtils
- Jakarta Commons Collections
- Jakarta Commons Digester
- Jakarta Commons Logging
- Mozilla Assert API

The JSF Specification requires JavaServer Pages Standard Tag Library (JSTL) as a dependency, therefore the required version of the JSTL from Jakarta is made available in the Application Server runtime.

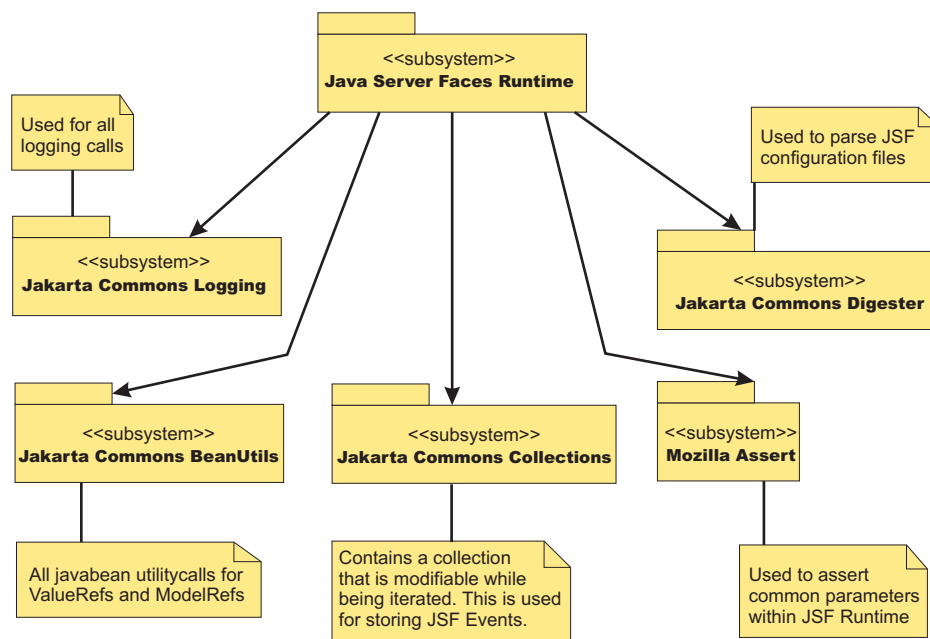
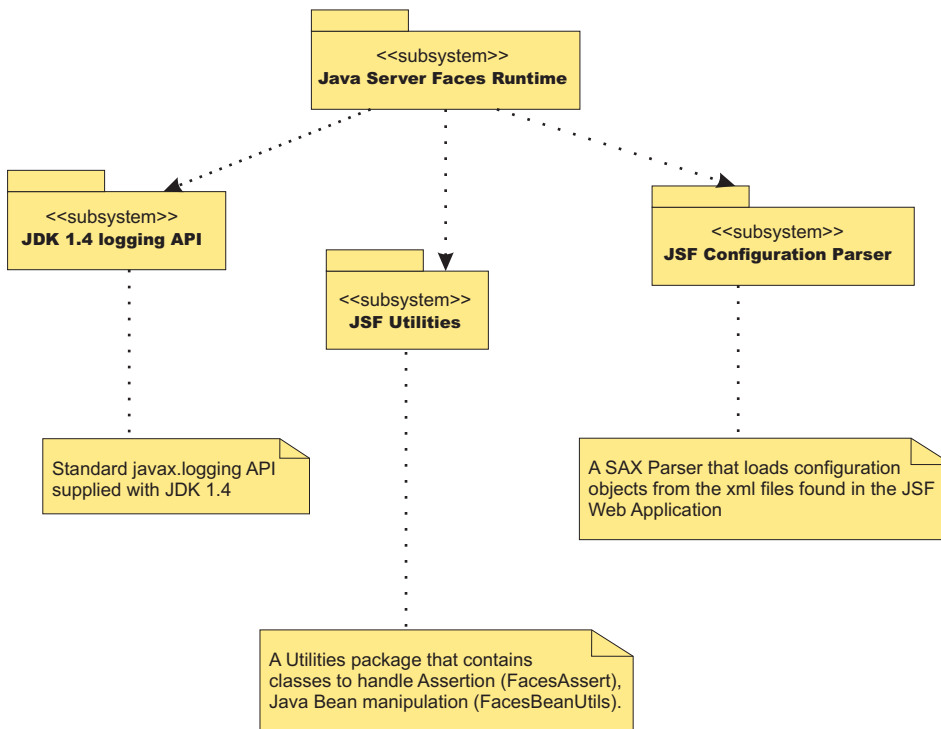


Figure 1. Current external API dependencies from the Sun based JSF runtime

Figure 2. Replacement APIs



The specification related classes (javax.faces.*) for JSF and the IBM modified version of the JSF Sun reference implementation are packaged in the Application Server runtime.

Typically Web applications that leverage this API/Framework embed the JSF API and implementation JAR files within their WAR file. This is not required when these Web applications are deployed and run within WebSphere Application Server. Only the removal of these jars along with any JSTL JAR files from the WAR file is required.

If a Web application requires the use of its own version of JSF or JSTL embedded within it, you can change the class loader mode of the Web application. By default this is set to PARENT_FIRST mode. Changing this value to PARENT_LAST allows the Web application version of the JSF or JSTL classes to load before the WebSphere Application Server.

FacesAssert class

The Sun Reference implementation uses a utility class from Mozilla to perform assertion style calls to method parameters. The faces assert class provides equivalent functionality. The option of leveraging the assertion functionality available in JDK 1.4 is not possible due to the requirement of providing JVM level parameters to turn on assertion code support. The FacesAssert class only contains static method and has no life cycle.

FacesAssert
+ notEmpty ([in] str : String) : boolean
+ nonNull ([in] isNull : Object) : boolean
+ wsAssert ([in] message : String) : boolean
+ wsAssert ([in] argument : boolean , [in] message : String) : boolean

FacesBeanUtils class

The FacesBeanUtils class provides static method replacements for methods used in the Jakarta Commons BeanUtils API. The FacesBeanUtils class has no life cycle.

FacesBeanUtils
+ getProperty ([in] bean : Object , [in] property : String) : Object
+ getPropertyType ([in] bean : Object , [in] property : String) : Class
+ getSimpleProperty ([in] bean : Object , [in] property : String , [in] value : Object)
+ getProperty ([in] bean : Object , [in] property : String , [in] value : Object)
+ convertFromString ([in] value : String , [in] valueClass : Class) : Object
+ convert ([in] targetType : Class , [in] bean : String) : Object

Faces configuration parser

The Sun Reference Implementation of JavaServer Faces use the Jakarta Commons Digester API to parse Faces configuration files. An XML SAX based parser is provided for the Application Server . The Digester code uses reflection code to perform its parsing. This has been found to be quite slow when large configuration files are parsed. The FaceConfigParser class in the diagram below is custom written for the Faces Configuration DTD and therefore parses large configuration files more quickly.



Figure 3. Faces configuration parser

Migrating Web application components from WebSphere Application Server Version 5.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.x and will override any J2EE 1.2 behaviors.

Servlet migration might be a concern if your application:

- implements a WebSphere Application Server 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.

- 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.x. You need to recompile all JSP pages when migrating from WebSphere Application Server Version 5.x.

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 4.0.1 EAR files (deployed in Version 4.0.1 with the JSP Precompile option), in Version 5.x. You need to recompile all JSP pages when migrating from WebSphere Application Server Version 5.x.

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

1. If a migrated application references internal servlets, enabled or disabled the functionality through the IBM WebSphere Extensions XML file, `ibm-web-ext.xml`, located in each Web module WEB-INF directory or by using assembly tools. Examples of this are `fileServing` and `serveServletsByClassName`.
2. Use WebSphere Application Server Version 5.x package names for any WebSphere Application Server Version 4.x internal servlets, which are implemented in your application.

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 open a Web deployment descriptor editor using an assembly tool and select **File serving enabled** on the **Extensions** tab.

3. Migrate servlets that extend `PageListServlet` and rely on configuration information in the associated XML servlet configuration file. In Version 4.0.1, 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.

HTTP session migration

There are no programmatic changes required to migrate from version 5.x to version 6.x. This article describes features that are available after migration.

Migration from Version 5.x

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

When you migrate between releases of WebSphere Application Server Version 5.x and later and you are using a database for session persistence, you can share the session database table between releases. For example, if you are accessing applications that are on WebSphere Application Server version 5.x you can share the session id with applications running on Version 6.x.

EJB applications

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.

- 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.

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.
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. For more information about migrating code in general, see "Resources for learning."

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. For more information about migrating code in general, see "Resources for learning."

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 specification 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.

Adjusting exception handling for EJB wrapped applications migrating from version 5 to version 6

Because of a change in the Java APIs for XML based Remote Procedure Call (JAX-RPC) specification, EJB applications that could be wrapped in WebSphere Application Server version 5.1 cannot be wrapped in version 6 unless you modify the code to the exception handling of the base EJB application.

Essentially, the JAX-RPC version 1.1 specification states:

a service specific exception declared in a remote method signature must be a checked exception. It must extend `java.lang.Exception` either directly or indirectly but it must not be a `RuntimeException`.

So it is no longer possible to directly use `java.lang.Exception` or `java.lang.Throwable` types. You must modify your applications using service specific exceptions to comply with the specification.

1. Modify your applications that use service specific exceptions. For example, say that your existing EJB uses a service specific exception called `UserException`. Inside of `UserException` is a field called `ex` that is type `java.lang.Exception`. To successfully wrapper your application with Web services in WebSphere Application Server version 6, you must change the `UserException` class. In this example, you could modify `UserException` to make the type of `ex` to be `java.lang.String` instead of `java.lang.Exception`.

new `UserException` class:

```
package irwwbase;

/**
 * Insert the type's description here.
 * Creation date: (9/25/00 2:25:18 PM)
 * @author: Administrator
 */

public class UserException extends java.lang.Exception {

    private java.lang.String _infostring = null;
    private java.lang.String ex;

    /**
     * UserException constructor comment.
     */

    public UserException() {
        super();
    }

    /**
     * UserException constructor comment.
     */
    public UserException (String infostring)
    {
        _infostring = infostring;
    } // ctor

    /**
     * Insert the method's description here.
     * Creation date: (11/29/2001 9:25:50 AM)
     * @param msg java.lang.String
     * @param ex java.lang.Exception
     */
    public UserException(String msg,String t) {
        super(msg);
        this.setEx(t);
    }
}
```



```

    }
    /**
     * @return
     */
    public java.lang.String get_infostring() {
        return _infostring;
    }

    /**
     * @return
     */
    public java.lang.String getEx() {
        return ex;
    }

    /**
     * @param string
     */
    public void set_infostring(java.lang.String string) {
        _infostring = string;
    }

    /**
     * @param Exception
     */
    public void setEx(java.lang.String exception) {
        ex = exception;
    }

    public void printStackTrace(java.io.PrintWriter s) {
        System.out.println("the exception is :"+ex);
    }
}

```

2. Modify all of the exception handling in the enterprise beans that use it. You must ensure that your enterprise beans are coded to accept the new exceptions. In this example, the code might look like this:

new EJB exception handling:

```

try {
    if (isDistributed()) itemCMPEntity = itemCMPEntityHome.findByPrimaryKey(ckey);
    else itemCMPEntityLocal = itemCMPEntityLocalHome.findByPrimaryKey(ckey);
} catch (Exception ex) {
    System.out.println("%%% ERROR: getItemInstance - CMPjdbc " + _className);
    ex.printStackTrace();
    throw new UserException("error on itemCMPEntityHome.findByPrimaryKey(ckey)",ex.getMessage());
}

```

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.

Interoperability of the handle formats in WebSphere Application Server, Version 5 and Version 5.0.1

Applications that attempt to persist handles to enterprise beans and **EJBHome** needed to subclass **ObjectInputStream** in WebSphere Application Server, Version 5. This action was required so that the subclass **ObjectInputStream** could utilize the context class loader to resolve the classes for enterprise beans and **EJBHome** stubs.

In addition, handles created and persisted in WebSphere Application Server, Version 5 only work with objects that have an unchanged remote interface. If the remote interface is changed, the handle is no longer valid because the stub is serialized inside the handle and its serial Version UID changes if the remote interface changes.

This release introduces a new handle persistence mechanism that avoids the implementation drawbacks of the previous version. However, if handles are used for this WebSphere Application Server deployment, you should consider the following issues when applying this update, future WebSphere Application Server Fix Packs and EJB Container cumulative fixes for WebSphere Application Server, Version 5.

If a WebSphere Application Server, Version 5 persisted handle or home handle is encountered by a WebSphere Application Server, Version 5.0.1 system, it can be read and utilized. In addition, it will be converted to WebSphere Application Server, Version 5.0.1 format if it is re-persisted. The WebSphere Application Server, Version 5.0.1 format cannot be read by a WebSphere Application Server, Version 5 system unless PQ72184 is applied.

Problems arise when handles are persisted and shared across systems that are not at the WebSphere Application Server, Version 5.0.1 level or later. However, a Version 5 system can receive a handle from Version 5.0.1 remotely through a call to get a handle on an enterprise bean or a getHomeHandle on an **EJBHome**. The remote call will succeed, however, any attempt to persist it on the Version 5 system will have the same limitations regarding the use of ObjectInputStream and changes in remote interface invalidating the persisted handle.

When your application stores handles persistently and shares this persistence with multiple clients or application servers, apply WebSphere Application Server, Version 5.0.1 or PQ72184 to both the client and server systems at the same time. Failure to do so can result in the inability of these systems to read the handle data stored by upgraded systems. Also, handles stored by the WebSphere Application Server, Version 5 can force the applications of the updated system to still subclass ObjectInputStream. Applications using the WebSphere Application Server Enterprise, Version 5 scheduler and process choreographer, are affected by these changes. These users should update their Version 5 systems at the same time with either Version 5.0.1 or PQ72184.

If the applications store handles in the session context, or locally in a file on the same system, that is not shared by other applications, on different systems, they might be able to update their systems individually, rather than all at once. If Client Container and thin client applications do not share persisted handle data, they can be updated as needed as well. However, handles created and persisted in WebSphere Application Server, Version 5, Version 4.0.3 and later (with the property flag set), or Version 3.5.7 and later (with the property flag set) are not usable if either the home or the remote interface changes.

If any WebSphere Application Server, Version 3.5.7 or Version 4.0.3 and later enables the system property `com.ibm.websphere.container.portable` to **true**, any handles to objects on that server have the same interoperability limitations. In addition, if any WebSphere Application Server, Version 3.5.7 and later or Version 4.0.3 applications store a handle obtained from a WebSphere Application Server, Version 5 or Version 5.0.1, the same restrictions apply, regarding the need to subclass ObjectInputStream and the usability of handles after a change to the remote interface is made.

Replication of the Http Session and Handles

This note applies to you if you place Handles to Homes or Enterprise JavaBeans, or EJB or EJBHome references in the Http Session in your application and you use Http Session Replication. If you intend to replicate a mixed environment of Version 5.0.0 and Version 5.0.1 or 5.0.2 machines you should first apply the latest Version 5.0.0 container cumulative e-fix to the Version 5.0.0 machines before allowing the Version 5.0.1 or 5.0.2 server into the typology. The reason for this is that Version 5.0.0 servers are not able to understand the persisted Handle format used on the Version 5.0.1 and 5.0.2 server. This is similar

to the case of Version 5.0.0 and Version 5.0.1 or 5.0.2 systems trying to use a shared database, mentioned above. But in this case, it is the Http Session object and not the database providing the persistence.

Top Down Deployment Mapping

The size of the Handle objects has grown due to the fix put in to allow serialization and deserialization to occur without the previous requirements of subclassing the ObjectInputStream and so on. Top down deployment of an object that contains EJB and EJBHome references create a database table ddl that has a field of 1000 bytes of VARCHAR for BITDATA which will contain the Handle. It might be that your object's Handle does not fit in the 1000 byte default field, and you might need to adjust this to a higher value. You might try increments of 250 bytes, that is, 1250, 1500, and so on.

Web services

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.

See how to modify your encoding from UTF-8 to UTF-16 if you need to change from UTF-8 to UTF-16.

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/wsdl/soap12-test"
  xmlns:types="http://whitemesa.net/wsdl/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.

How to change encoding in a SOAP message to support WSI-Basic Profile

Support for Universal Transformation Format (UTF)-16 encoding is required by the WS-I Basic Profile 1.0. WebSphere Application Server conforms to the WS-I Basic Profile 1.1. 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 and UTF-16 encoding is typically used for Java and Windows product applications. This task teaches you how to change the encoding in a SOAP message from UTF-8 to UTF-16.

To learn more about the requirements of the Web Services-Interoperability Basic Profile (WS-I), including UTF-16, see [Web Services-Interoperability Basic Profile](#).

Support for UTF-16 encoding is required by WS-I Basic Profile; therefore, you need to know how to modify your character encoding from UTF-8 to UTF-16 in your SOAP message.

You can change the character encoding in one of two ways:

- Use a property on the Stub for users to set.

This choice applies to the client only.

For a client, the encoding is specified in the SOAP request. The SOAP engine serializes the request and sends it to the Web service engine. The Web service engine receives the request and deserializes the message to Java objects, which are returned to you in a response.

When the Web service engine on the server receives the serialized request, a raw message in the form of an input stream, is passed to the parser, which understands Byte Order Mark (BOM). BOM is mandatory for UTF-16 encoding and it can be used in UTF-8. The message is deserialized to a Java objects and a service invocation is made. For two-way invocation, the engine needs to serialize the

message using a specific encoding and send it back to the caller. The following example shows you how to use a property on the Stub to change the character set:

```
javax.xml.rpc.Stub stub=service.getPort("MyPortType");
((javax.xml.rpc.Stub)stub).setProperty(com.ibm.wsspi.webservices.Constants.MESSAGE_CHARACTER_SET_ENCODING,"UTF-16");
stub.invokeMethod();
```

In this code example, `com.ibm.wsspi.webservices.Constants.MESSAGE_CHARACTER_SET_ENCODING = "com.ibm.wsspi.webservices.xmlcharset"`;

- Use a handler to change the character set through SOAP with Attachments API for Java (SAAJ).

If you are using a handler, the SOAP message is transformed to a SAAJ format from other possible forms, such as an input stream. In such cases as a `handleRequest` method on the client side and a `handleResponse` method on the server side, the Web services engine transforms from a SAAJ format back to the stream with appropriate character encoding. This transformation or change is called a *roundtrip transformation*. The following is an example of how you would use a handler to specify the character encoding through SAAJ:

```
handleResponse(MessageContext mc) {
    SOAPMessageContext smc = (SOAPMessageContext) context;
    javax.xml.soap.SOAPMessage msg = smc.getMessage();
    msg.setProperty(javax.xml.soap.SOAPMessage.CHARACTER_SET_ENCODING, "UTF-16");
}
}
```

You have modified the character encoding from UTF-8 to UTF-16 in the Web service SOAP message.

Migrating Web services

This topic explains information that you need to migrate Web services.

If you have used the Apache SOAP support to develop Web services client applications in WebSphere Application Server Versions 4, 5, or 5.1, you might need to migrate your applications or the security files for your applications. The following table summarizes the Web services specifications supported by the WebSphere products.

Table 1.

WebSphere Application Server Version	Web services specifications supported
4.0	SOAP 2.2
5.0 and 5.0.1	SOAP 2.3
5.0.2 or later	Java 2 Platform, Enterprise Edition (J2EE), also known as (JSR 109)
6.0	J2EE (JSR 109)

The SOAP 2.2 and SOAP 2.3-based implementations that were available in WebSphere Application Server Version 4.0.x, 5.0 and 5.0.1 have been deprecated. It is recommended that applications that are using these SOAP implementations migrate to Web Services for J2EE (JSR 109) support that is provided in current WebSphere Application Server versions.

For more information on migrating your Web services, see [Migrating Apache SOAP Web services to Web Services to J2EE standards](#) and [Web services migration best practices](#).

It is recommended that new Web services be developed using the Web services for J2EE specification. For more information, see the article on [implementing Web services applications](#) in the information center.

Security cannot be directly migrated from SOAP 2.3 to the J2EE standards. After you have migrated your Web services to the J2EE standards, see the article on [securing Web services for Version 6 applications based on WS-Security](#) in the information center.

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, see the article on developing a Web services client based on Web Services for J2EE in the information center.
 - 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. See the article in the information center on how to develop a Web services client based on the Web Services for J2EE specification.

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

Web services migration best practices

This topic presents best practices for migrating Web services applications.

Migrating a Version 5 Java API for XML-based remote procedure call (JAX-RPC) client that uses SOAP over Java Message Service (JMS) to invoke a Web service

A JAX-RPC client that is executed on WebSphere Application Server Version 5, can use SOAP over JMS to invoke a Web service that is executed on a Version 5 application server.

A user ID and password are not required on the target WebSphere MQ queue. After the application server is migrated to Version 6.x, and uses the Version 6.x default messaging feature, client requests can fail because basic authentication is enabled. The following error message displays when this migration problem occurs:

```
SibMessage W [:] CWSIT0009W: A client request failed in the application server with
endpoint <endpoint name> in bus <bus_name> with reason: CWSIT0016E: The user
ID null failed authentication in bus <bus_name>.
```

When the application server is migrated to Version 6.x, and the default messaging provider (service integration technologies) is used, and administrative and application security is enabled for the server or the cell, the service integration bus queue destination inherits the security characteristics of the server or the cell by default. If the server or the cell has basic authentication enabled, the client request fails.

The following options are available to solve this problem. The solutions are listed by the level of security that they impose:

- Disable administrative and application security on the main security panel within the administrative console. To disable administrative and application security, click **Security > Secure administration, applications and infrastructure**. Deselect the **Enable administrative security** and **Enable application security** options.
- Modify the settings for the service integration bus that hosts the queue destination so that the bus security is disabled and the bus does not inherit security characteristics from the server or the cell. This option is equivalent to the level of security that you can configure in Version 5.
- Configure the basic authentication on each client that uses the service.

Migrating Apache SOAP Web services

See Migrating Apache SOAP Web services to Web Services for J2EE standards to learn how to migrate Apache SOAP Web services. 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.

Migrating Web services assembled with early versions of the Application Server Toolkit or Assembly Toolkit

If you are migrating your Web service or Web service components from earlier versions of the Application Server Toolkit or Assembly Toolkit, refer to the following hints and tips to improve your success:

- Secure Web services are not migrated by the J2EE Migration Wizard when Web services are migrated from J2EE 1.3 to J2EE 1.4.
- The migration of secure Web services requires manual steps.
- After the J2EE migration, the secure binding and extension files must be migrated manually to J2EE 1.4 as follows:
 1. Double click on the `webservices.xml` file to open the Web Services editor.
 2. Select the **Binding Configurations** tab to edit the binding file.
 3. Add all the necessary binding configurations under the new sections **Request Consumer Binding Configuration Details** and **Response Generator Binding Configuration Details**.
 4. Select the **Extension** tab to edit the extension file.
 5. Add all the necessary extension configurations under the new sections **Request Consumer Service Configuration Details** and **Response Generator Service Configuration Details**.
 6. Save and exit the editor.

Migrating the UDDI registry

With most scenarios, migration of existing UDDI registries happens automatically when you migrate to the current level of WebSphere Application Server. However, if your existing UDDI registry uses a network Cloudscape database or a DB2 UDDI Version 2 database, there are some manual steps that you must take.

Migrate your installation of WebSphere Application Server; ensure that you select the option to migrate applications, so that the UDDI registry application will be migrated.

If your existing UDDI registry uses an Oracle, embedded Cloudscape or DB2 UDDI Version 3 database, you do not need to perform any manual migration; migration happens automatically when you migrate WebSphere Application Server and start the UDDI node for the first time after migration.

If your existing UDDI registry uses a network Cloudscape database or a DB2 UDDI Version 2 database, you must complete some manual steps to migrate the registry.

- If your UDDI registry uses a DB2 UDDI Version 2 database, follow the steps in “Migrating to Version 3 of the UDDI registry” on page 44 and sub-topics.
- If your UDDI registry uses a network Cloudscape database, complete the following steps.
 1. If you have a cluster that contains servers at different levels of WebSphere Application Server, ensure that any UDDI registries are running on servers that are at WebSphere Application Server Version 6.1. For example, if you have a cluster that spans two nodes, you can upgrade one node to WebSphere Application Server Version 6.1 while the other node remains at a previous level, provided that any servers that are running a UDDI registry are at Version 6.1.
 2. Initialize the relevant UDDI node. The initialize process will perform some of the UDDI registry migration.

3. Enter the following commands as the database administrator, from *app_server_root/cloudscape/lib*.

```
java -cp db2j.jar;db2jtools.jar com.ibm.db2j.tools.ij
connect 'jdbc:db2j:uddi_cloudscape_database_path';
run 'app_server_root/UDDIReg/databaseScripts/uddi30crt_drop_triggers_cloudscape.sql';
quit;
cd app_server_root/derby/migration
java -cp db2j.jar;db2jmigration.jar;../lib/derby.jar com.ibm.db2j.tools.MigrateFrom51
  jdbc:db2j:uddi_cloudscape_database_path
```

where

- *uddi_cloudscape_database_path* is the absolute path of the existing Cloudscape database, for example *app_server_root/profiles/profile_name/databases/com.ibm.uddi/UDDI30*
- *app_server_root* is the root directory for the installation of WebSphere Application Server

The UDDI database and data source are migrated, and the UDDI node is activated.

Note: When you migrate WebSphere Application Server, the post-upgrade log for the profile indicates that the migration of the UDDI database is partially complete, and is missing the steps for triggers, aliases, and stored statements. If you initially enabled the debug function, the debug log for the database indicates that there was a failure creating triggers. Ignore these messages; the UDDI node completes the migration of the database when the UDDI node starts. For more information about these log files, see “Verifying the Cloudscape v10.1.x automatic migration” on page 50. Also refer to this topic if other errors appear in the logs.

If the migration of the UDDI database completes successfully, the following message appears in the server log:

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

If the following error appears, an unexpected error occurred during migration. The UDDI registry node is not activated. Check the error logs for the problem and, if it cannot be fixed, see the IBM software support Web site at <http://www.ibm.com/software/support>.

```
CWUDQ004W: UDDI registry not started due to migration errors
```

Migrating to Version 3 of the UDDI registry

Use this topic to migrate a Version 2 UDDI registry that uses a DB2 database, running in WebSphere Application Server Version 5 or later, to a Version 3 UDDI registry running in WebSphere Application Server Version 6.x.

The following constraints apply to this procedure:

- Your existing registry must use a DB2 database.
- Your existing registry must run in WebSphere Application Server Version 5 or later. If you are migrating from UDDI registry Version 1.1 or 1.1.1, which ran on WebSphere Application Server Version 4, 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 must have already migrated the WebSphere Application Server Version 5 deployment manager to Version 6.x.

You can only run the Version 2 UDDI registry, supplied in WebSphere Application Server Version 5, in a WebSphere Application Server Version 6.x server if you are running in a mixed cell migration mode. In this

configuration, the WebSphere Application Server Version 6.x deployment manager manages the version 5.x nodes, and the Version 2 UDDI registry is supported only for migration purposes; it is not supported for normal execution.

If you have the UDDI registry deployed in a clustered application server and you migrate to WebSphere Application Server Version 6.x, you will not be able to run UDDI across the mixed-version cluster. You can continue to run the UDDI registry on the server or servers that remain at Version 5 of WebSphere Application Server, but you will not be able to run the UDDI registry in the Version 6.x servers of WebSphere Application Server until all nodes in the cluster have been migrated to WebSphere Application Server Version 6.x. 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.x Network Deployment configuration, a number of choices are available, including having a mixed version cell where the WebSphere Application Server version 6.x 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.

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/cell_name/nodes/node_name/servers/server_name` 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 the server from WebSphere Application Server Version 5.x to Version 6.x. This results in a new directory tree for the migrated Version 6.x application server.
4. Start the new, migrated Version 6.x application server.
5. Create a new data source for the Version 2 UDDI database. This data source is known as the *UDDI migration data source*. The JNDI name must be `datasources/uddimigration`. To complete this step see *Setting up a UDDI migration datasource*.
6. Set up the UDDI Version 3 registry and migrate the Version 2 data.

Follow the instructions in the topic in the information center on setting up a customized UDDI node, including the subtopic that relates to node initialization. The topic describes how to perform the following actions:

- Create the Version 3 DB2 database
- Create the J2C authentication data entry
- Create the JDBC provider and data source
- Deploy the UDDI registry application
- Start the server
- Configure and initialize the node. The UDDI registry node initialization detects the UDDI migration data source, and migrates the Version 2 data as part of the UDDI node initialization processing. This data migration can take some time, depending on the amount of data in your UDDI registry.

The UDDI registry is migrated. If the UDDI node remains in a state of initialization pending, migration pending or value set creation pending, check the server log for errors. If the following message appears, an unexpected error occurred during migration. Check the error logs for the problem and, if it cannot be fixed, see the IBM software support Web site at <http://www.ibm.com/software/support>.

CWUDQ004W: UDDI registry not started due to migration errors

After the problem is fixed, you can complete the migration by clicking **Initialize** again.

Verify that the migration process completed successfully by checking for the following message in the server log:

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

After migration is complete you can remove the UDDI migration data source, and the registry is available for use.

Setting up a UDDI migration datasource

Use this topic 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 JDBC Provider for DB2 does not already exist, then create one, selecting the options DB2 Universal JDBC Driver Provider and Connection Pool datasource.

For details on how to create a JDBC provider, see the article in the information center on creating and configuring a JDBC provider using the administrative console.

2. Create a datasource for the Version 2 UDDI registry by following these steps:

- a. Expand **Resources** and **JDBC Providers**.

- b. Click **Resources** → **JDBC** → **JDBC Providers**.

- c. Select the desired 'scope' of the JDBC provider you selected or created earlier. For example, select:

Server: yourservername

to show the JDBC providers at the server level.

- d. Select the JDBC provider created earlier.

- e. Under **Additional Properties**, select **Data sources** (*not* the **Data sources (WebSphere Application Server V4)** option).

- f. Click **New** to create a new datasource.

- g. In the **Create a data source** wizard, enter the following data:

Name a suitable name, such as UDDI Datasource

JNDI name

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

Component-managed authentication alias

select the alias for the DB2 userid used to access UDDI Version 2 data, for example MyNode/UDDIAlias

- h. Click **Next**.

- i. On the database specific properties page of the wizard, enter the following data:

Database name

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

Use this Data Source in container-managed persistence (CMP)

ensure the check box is cleared.

- j. Click **Next**, then check the summary and click **Finish**.

- k. Click the data source to display its properties, and add the following information:

Description

a suitable description

Category

set to **uddi**

Data store helper class name

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

Mapping-configuration alias

set to DefaultPrincipalMapping

3. Click **Apply** and save the changes to the master configuration.
4. Test the connection to your UDDI database by selecting the check box next to the datasource and clicking **Test connection**. You will see a message similar to "Test Connection for datasource UDDI Datasource on server server1 at node MyNode was successful". If you do not see this message investigate the problem with the help of the error message.

Continue with the migration as detailed in Migrating to Version 3 of the UDDI registry.

Data access resources

Migrating applications to use data sources of the current J2EE Connector Architecture (JCA)

Migrate your applications that use Version 4 data sources, or data sources (WebSphere Application Server V4), to use data sources that support more advanced connection management features, such as connection sharing.

To use the connection management infrastructure in WebSphere Application Server Version 6.x, 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.

Applications left at the J2EE 1.2 level will continue to run fine using the connection management support that was available at V4.0; simply create the JDBC provider and data source, and install the 4.0 application as-is. If you choose to repackage your application for version 6.0 or above you can not use a Version 4.0 data source. You must use the other data source option, which supports applications coded to the J2EE 1.3 specifications, at minimum.

In WebSphere Application Server Version 6.x, data sources are intended for use within J2EE applications and designed to operate within the EJB and Web containers.

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 Creating a resource reference article in the information center.
 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 Creating a resource reference article in the information center.
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. Install the application following the instructions in the Installing a new application article in the information center, and bind the resource references to the data source that you created.
2. Perform the necessary administrative task of creating a JDBC provider and a data source object following the instructions in the Creating a JDBC provider and data source article in the information center.

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-sharable 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 the `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). See the Examples section for a description of call nesting.

Verifying the Cloudscape v10.1.x automatic migration

WebSphere Application Server Version 6.1.x requires Cloudscape to run at a minimal version of v10.1.x. (Note that Cloudscape v10.1.x is comprised of the Derby code base.) During the Application Server v6.1.x upgrade, the migration tool automatically upgrades the database instances that are accessed through the

embedded framework by some internal components, such as the UDDI registry. The tool also attempts to upgrade Cloudscape instances that your applications access through the embedded framework. You must verify the migration results for these backend databases.

Do not use Cloudscape v10.1.x as a production database. Use it for development and test purposes only.

Learn more: The new version of Cloudscape combines the Derby runtime with additional benefits, such as IBM Quality Assurance (QA) and national language support (NLS). For information about the Cloudscape v10.1.x open source code base, see the Cloudscape section of **ibm.com**: <http://www-306.ibm.com/software/data/cloudscape/>.

The migration tool attempts to upgrade Cloudscape database instances that are accessed through the embedded framework only. You must manually upgrade Cloudscape instances that transact with application servers on the Network Server framework. (See the “Upgrading Cloudscape manually” on page 53 article.) This requirement eliminates the risk of corrupting third party applications that use the Network Server framework to access the same database instances as WebSphere Application Server.

Cloudscape requires the Network Server framework to transact with more than one Java Virtual Machine (JVM) concurrently. Within WebSphere Application Server, each application server is a JVM; hence you can use a single instance of Cloudscape Network Server framework with a clustered or coexistence implementation of Application Server. For more information on the Network Server framework, consult the IBM Cloudscape information center at <http://publib.boulder.ibm.com/infocenter/cscv/v10r1/index.jsp>.

For database instances that your applications access through the embedded framework, the automatic migration can succeed completely, fail completely, or succeed with warnings. A migration that produces warning messages does create a Cloudscape v10.1.x database with your data, but does not migrate all of your configured logic and other settings, such as:

- keys
- checks
- views
- triggers
- aliases
- stored statements

To distinguish between a partially and a completely successful migration, you must verify the auto-migration results by checking both the general post-upgrade log and the individual database logs. Performing these tasks gives you vital diagnostic data to troubleshoot the partially migrated databases as well as those that fail auto-migration completely. Ultimately, you migrate these databases through a manual process.

1. Open the post-upgrade log of each new WebSphere Application Server Version 6.1x profile. The path name of the log is `WAS_HOME/profiles/profileName/logs/WASPostUpgrade.timestamp.log`.
2. Examine the post-upgrade log for database error messages. These exceptions indicate database migration failures. The following lines are an example of post-upgrade log content, in which the database error code is DSRA7600E. (The migration tool references all database exceptions with the prefix DSRA.)

```
MIGR0344I: Processing configuration file /opt/WebSphere51/AppServer/cloudscape
/db2j.properties.
```

```
MIGR0344I: Processing configuration file /opt/WebSphere51/AppServer/config/cells
/migr06/applications/MyBankApp.ear/deployments/MyBankApp/deployment.xml.
```

```
DSRA7600E: Cloudscape migration of database instance /opt/WebSphere61/Express
/profiles/default/databases/_opt_WebSphere51_AppServer_bin_DefaultDB failed,
reason: java.sql.SQLException: Failure creating target db
```

MIGR0430W: Cloudscape Database /fvt/temp/51BaseXExpress/PostUpgrade50BaseFVTTTest9 /testRun/pre/websphere_backup/bin/DefaultDB failed to migrate <new database name>

Important: Call IBM WebSphere Application Server Support if you see a migration failure message for a Cloudscape instance that is accessed by a WebSphere internal component (that is, a component that helps comprise WebSphere Application Server rather than one of your applications).

- Open the individual database migration log that corresponds with each of your backend Cloudscape databases. These logs have the same timestamp as that of the general post-upgrade log. The logs display more detail about errors that are listed in the general post-upgrade log, as well as expose errors that are not documented by the general log.

The path name of each database log is `WAS_HOME/profiles/profileName/logs/myFullDbPathName_migrationLogtimestamp.log`.

- Examine each database migration log for errors. For a completely successful migration, the log displays a message that is similar to the following text:

```
MIGR0429I: Cloudscape Database F:\temp\51BaseXExpress\PostUpgrade50BaseFVTTTest2\testRun
\pre\websphere_backup\bin\DefaultDB was successfully migrated. See log C:\WebSphere61
\Express\profiles\default\logs\DefaultDB_migrationLogSun-Dec-18-13.31.40-CST-2005.log
```

Otherwise, the log displays error messages in the format of the following example:

```
connecting to source db <jdbc:db2j:/fvt/temp/51BaseXExpress/PostUpgrade50BaseFVTTTest9
/testRun/pre/websphere_backup/bin/DefaultDB>
```

```
connecting to source db <jdbc:db2j:/fvt/temp/51BaseXExpress/PostUpgrade50BaseFVTTTest9
/testRun/pre/websphere_backup/bin/DefaultDB> took 0.26 seconds
```

```
creating target db <jdbc:derby:/opt/WebSphere61/Express/profiles/default/databases
/_opt_WebSphere51_AppServer_bin_DefaultDB>
```

```
ERROR: An error occurred during migration. See debug.log for more details.
```

```
shutting down databases
```

```
shutting down databases took 0.055 seconds
```

- For more data about a migration error, consult the debug log that corresponds with the database migration log. The WebSphere Application Server migration utility triggers a *debug migration trace* by default; this trace function generates the database debug logs. The full path name of a debug log is `WAS_HOME/profiles/profileName/logs/myFullDbPathName_migrationDebugtimestamp.log`.

The following lines are a sample of debug text. The lines display detailed exception data for the error that is referenced in the previous sample of database migration log data.

```
java.sql.SQLException: Database_opt_WebSphere51_AppServer_bin_DefaultDB already exists. Aborting migration
at com.ibm.db2j.tools.migration.MigrateFrom51Impl.go(Unknown Source)
at com.ibm.db2j.tools.migration.MigrateFrom51Impl.doMigrate(Unknown Source)
at com.ibm.db2j.tools.MigrateFrom51.doMigrate(Unknown Source)
at com.ibm.ws.adapter.migration.CloudscapeMigrationUtility.migr
```

- The WebSphere Application Server migration utility changes your Cloudscape JDBC configurations whether or not it successfully migrates the database instances that are accessed by your applications. The tool changes Cloudscape JDBC provider class paths, data source implementation classes, and data source helper classes. The following table depicts these changes:

Class type	Old value	New value
------------	-----------	-----------

JDBC provider class path	<code>\${CLOUDSCAPE_JDBC_DRIVER_PATH}/db2j.jar</code>	<code>\${DERBY_JDBC_DRIVER_PATH}/derby.jar</code> <ul style="list-style-type: none"> • Where <code>DERBY_JDBC_DRIVER_PATH</code> is the WebSphere environment variable that defines your Cloudscape JDBC provider • Where <code>derby.jar</code> is the base name of the JDBC driver class file (In your environment, reference the JDBC driver class file by the full path name.)
Data source implementation class: Connection pool	<code>com.ibm.db2j.jdbc.DB2jConnectionPoolDataSource</code>	<code>org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource</code>
Data source implementation class: XA	<code>com.ibm.db2j.jdbc.DB2jXADataSource</code>	<code>org.apache.derby.jdbc.EmbeddedXADataSource</code>
Data source helper class	<code>com.ibm.websphere.rsadapter.CloudscapeDataStoreHelper</code>	<code>com.ibm.websphere.rsadapter.DerbyDataStoreHelper</code>

Additionally, the `db2j.properties` file changes:

- The name `WAS_HOME/cloudscape/dbj.properties` changes to `WAS_HOME/derby/derby.properties`
- Within the file, property names change from `db2j.drda.*` to `derby.drda.*`
- A partial or a completely successful database migration changes the location and name of the database according to the following example:
 - Old database name: `c:\temp\mydb`
 - New database name: `WAS_HOME\profiles\profilename\databases\c_temp_mydb`

Note the exact path names: For both partial and failed migrations, the log messages contain the exact old and new database path names that you must use to run the manual migration. Note these new path names precisely.

If you experience a partial migration, attempt to troubleshoot the new v10.1.x database only if you have expert knowledge of Cloudscape. Otherwise, delete the new database. Perform the manual migration procedure on the original database, just as you do for each database that completely fails auto-migration. Consult “Upgrading Cloudscape manually” for instructions.

For successfully migrated Cloudscape instances, be aware that new cell-scoped data sources can only be used by nodes that run WebSphere Application Server v6.02 or later. Earlier versions of the product do not support the new Cloudscape; when applications on pre-v6.02 nodes try to access a Cloudscape 10.1.x data source, Application Server issues run-time exceptions.

Upgrading Cloudscape manually

During the WebSphere Application Server v6.1.x upgrade, the migration tool attempts to upgrade instances of Cloudscape that are accessed through the embedded framework only. (The new version of Cloudscape is version 10.1.x, which is based on Derby.) The automatic upgrade excludes Cloudscape instances that transact with applications through the Network Server framework. This exclusion eliminates the risk of corrupting third party applications that access the same database instances as WebSphere Application Server. You must manually upgrade database instances that are accessed through the Network Server framework. Do the same for databases that fail the automatic migration.

Do not use Cloudscape v10.1.x as a production database. Use it for development and test purposes only.

Learn more: The new version of Cloudscape combines the Derby runtime with additional benefits, such as IBM Quality Assurance (QA) and national language support (NLS).

- For information about the Cloudscape v10.1.x open source code base, see the Cloudscape section of [ibm.com](http://www-306.ibm.com/software/data/cloudscape/): <http://www-306.ibm.com/software/data/cloudscape/>.
- You can investigate incompatibilities between Cloudscape v10.1.x and v5.1.60x (plus versions prior to v5.1.60x) by consulting the Cloudscape migration document at <http://publib.boulder.ibm.com/epubs/html/c1894711.html>.

For instances of Cloudscape that are accessed through the embedded framework, determine which instances completely failed the automatic upgrade process and which ones were only partially upgraded. The article “Verifying the Cloudscape v10.1.x automatic migration” on page 50 documents how to uncover database errors and diagnostic data from various migration logs. The log messages contain the exact old and new database path names that you must use to run the manual migration. Note these new path names precisely.

To minimize the risk of migration errors for databases that were only partially upgraded during the automatic migration process, delete the new database. Troubleshoot the original database according to the log diagnostic data, then perform manual migration on the original database.

The following section consists of steps to migrate Cloudscape instances that are accessed through both frameworks: the embedded as well as the Network Server framework. Steps that apply only to the Cloudscape Network Server framework are marked accordingly. As a migration best practice, ensure that your user ID has one of the following authorities:

- Administrator of the application server that accesses the Cloudscape instance
- A umask that can access the database instance

Otherwise, you might see runtime errors about the database instance being read-only.

1. **Network Server framework only:** Ensure that every client of the Cloudscape database can support Cloudscape v10.1.x. WebSphere Application Server clients of the database must run versions 6.1.x or 6.02.x of Application Server.

In the case of mixed-node cells, remember that only nodes of WebSphere Application Server v6.02 or later can use data sources that you create post-migration for access to Cloudscape 10.1.x. Earlier versions of the product do not support the new Cloudscape; when applications on pre-v6.02 nodes try to access a cell-scoped Cloudscape 10.1.x data source, Application Server issues run-time exceptions.

2. **Network Server framework only:** Take the database offline. No clients can access it during the migration process.
3. Examine a sample Cloudscape migration script that Application Server provides: either `db2jmigrate.bat`, or `db2jmigrate.sh`. The path of both scripts is `WAS_HOME\derby\bin\embedded\...`. You can modify the script according to the requirements of your environment. Consult the Cloudscape migration document at <http://publib.boulder.ibm.com/epubs/html/c1894711.html> for information about options that you can use with the script. For example, you can use the option `-DB2j.migrate.ddlFile=filename` to specify the DDL file for the new database.
4. To generate database debug logs when you run the migration script, ensure that the *debug migration trace* is active. By default, this trace function is enabled. To reactivate the debug trace if it is disabled, set one of the following trace options:
 - `all traces*=all`
 - `com.ibm.ws.migration.WASUpgrade=all`
5. Specify your old database name and the full post-migration path of the new database name when you run the script. For example: `E:\WebSphere\AppServer\derby\bin\embedded>db2jMigrate.bat myOldDB myNewDB` The logs from the automatic migration provide the exact path names to specify for both the old database and the target database. You must use this target database name to specify the new database, because your migrated Cloudscape data sources (updated by the WebSphere Application Server migration utilities) now point to the target database name. The following sample text demonstrates how log messages display target database names:

```
DSRA7600E: Cloudscape migration of database instance C:\temp\migration2\profiles\AppSrv01\
installedApps\ghongellNode01Cell\DynamicQuery.ear\EmployeeFinderDB to new database instance
C:\WebSphere\AppServer\profiles\AppSrv01\databases\C_WAS602_AppServer_profiles_AppSrv01_
installedApps_ghongellNode01Cell_DynamicQuery.ear_EmployeeFinderDB failed,
reason: java.sql.SQLException: Failure creating target db
```

For instances of Cloudscape that are accessed through the Network Server framework, input any name that you want for the new database. Remember to modify your existing data sources to point to the new database name.

6. When the migration process ends, examine the database migration log to verify the results. The path name of each database migration log is `WAS_HOME/logs/derby/myFullDbPathName_migrationLog.log`. For a successful migration, the database migration log displays a message that is similar to the following text:

```
Check E:\WebSphere\AppServer\derby\my01dDB_migrationLog.log for progress
Migration Completed Successfully
E:\WebSphere\AppServer\derby\bin\embedded>
```

Otherwise, the log displays error messages in the format of the following example:

```
Check E:\WebSphere\AppServer\derby\my01dDB_migrationLog.log for progress
ERROR: An error occurred during migration. See debug.log for more details.
ERROR XMG02: Failure creating target db
java.sql.SQLException: Failure creating target db
    at com.ibm.db2j.tools.migration.MigrationState.getCurrSQLException(Unknown Source)
    at com.ibm.db2j.tools.migration.MigrateFrom51Impl.handleException(Unknown Source)
    at com.ibm.db2j.tools.migration.MigrateFrom51Impl.go(Unknown Source)
    at com.ibm.db2j.tools.migration.MigrateFrom51Impl.main(Unknown Source)
    at com.ibm.db2j.tools.MigrateFrom51.main(Unknown Source)
```

...

7. For more data about a migration error, consult the debug log that corresponds with the database migration log. The full path name of a debug log file is `WAS_HOME/logs/derby/myFullDbPathName_migrationDebug.log`

The following lines are a sample of debug text.

```
sourceDBURL=jdbc:db2j:E:\WebSphere\my01dDB
newDBURL=jdbc:derby:e:\tempo\myNewDB
ddlOnly=false
connecting to source db <jdbc:db2j:E:\WebSphere\my01dDB>
connecting to source db <jdbc:db2j:E:\WebSphere\my01dDB> took 0.611 seconds
creating target db <jdbc:derby:e:\tempo\myNewDB>
creating target db <jdbc:derby:e:\tempo\myNewDB> took 6.589 seconds
initializing source db data structures
initializing source db data structures took 0.151 seconds
recording DDL to create db <E:\WebSphere\my01dDB>
recording DDL to create db <E:\WebSphere\my01dDB> took 5.808 seconds
```

As indicated in the previous steps, the database migration log displays either a Migration Completed Successfully message, or a message containing migration failure exceptions.

- For databases that fail migration, troubleshoot according to the logged error data. Then rerun the migration script.
- To access successfully upgraded databases through the embedded framework, modify your data sources to point to the new database names.
- To access successfully upgraded databases through the Network Server framework, you can use either the DB2 Universal JDBC driver or the Derby Client JDBC driver.
 - If you want your existing JDBC configurations to continue to use the DB2 Universal JDBC driver, modify your data sources to point to the new database names.
 - If you want to use the Derby Client JDBC driver, which can support XA data sources, modify your JDBC providers to use the new Derby Client JDBC driver class and the new data source implementation classes. Then reconfigure every existing data source to use the correct Derby data source helper class, and to point to the new database name.

Consult the article in the information center on vendor-specific data sources minimum required settings for all of the new class names.

Security

Migrating, coexisting, and interoperating – Security considerations

Use this topic to migrate the security configuration of previous WebSphere Application Server releases and its applications to the new installation of WebSphere Application Server.

This information addresses the need to migrate your security configurations from a previous release of IBM WebSphere Application Server to WebSphere Application Server Version 6.1 or later. Complete the following steps to migrate your security configurations:

- If security is enabled in the previous release, obtain the administrative server ID and password of the previous release. This information is needed in order to run certain migration jobs.
- You can optionally disable security in the previous release before migrating the installation. No logon is required during the installation.
- If `scriptCompatibility` is `false` when migrating to WebSphere Application Server 6.1 on z/OS, any SSLConfig repertoire of type System SSL (SSSL) is converted to type JSSE. The exception is when the SSLConfig repertoire belongs to the daemon; the repertoire is not converted from type SSSL to type JSSE in this case.

Migrate the appropriate product configuration. You can migrate the base application server node, a deployment manager, and a federated node.

- To migrate a base application server node, see *Migrating stand-alone application servers*.
- To migrate a deployment manager, see *Migrating deployment managers*.
- To migrate a federated node, see *Managed federate nodes*.

The security configuration of previous WebSphere Application Server releases and its applications are migrated to the new installation of WebSphere Application Server Version 6.1.

If a custom user registry is used in the previous version, the migration process does not migrate the class files that are used by the standalone custom registry in the previous `app_server_root/classes` directory. Therefore, after migration, copy your custom user registry implementation classes to the `app_server_root/classes` directory.

If you upgrade from WebSphere Application Server, Version 5.x to WebSphere Application Server, Version 6.1, the data that is associated with Version 5.x trust associations is not automatically migrated to Version 6.1. To migrate trust associations, see “Migrating trust association interceptors” on page 61.

If you are migrating a Version 5.x or 6.0.x environment with Sync to OS Thread enabled to a Version 6.1 environment, you should be aware of the following migration considerations:

- In addition to the application and configuration specifying the desire to use Sync to OS Thread that was required in earlier versions of WebSphere Application Server, the RACF administrator must also define a resource role in order for Sync to OS Thread to operate in Version 6.1 and later. A FACILITY class profile must be defined to allow or disallow the use of Sync to OS Thread. Also, an optional SURROGAT class profile can be used to further refine the use of Sync to OS Thread to particular authenticated users.

See the *Securing applications and their environment* PDF.

- In Version 6.1 and later, a FACILITY class profile must be defined to enable trust applications. WebSphere Applications Server checks this FACILITY class profile during initialization to ensure that only authorized trusted applications are enabled. This FACILITY class profile expands the RACF administrator’s role in ensuring that only authorized trusted applications are enabled.

See the *Securing applications and their environment* PDF.

Interoperating with previous product versions

IBM WebSphere Application Server interoperates with the previous product versions. Use this topic to configure this behavior.

1. Configure WebSphere Application Server Version 6.1 with the same distributed user registry (that is, LDAP or Custom) that is configured with the previous version. Make sure that the same LDAP user registry is shared by all of the product versions.
 - a. In the administrative console, select **Security > Secure administration, applications, and infrastructure**.
 - b. Choose an available Realm definition and click **Configure**.
 - c. Enter a **Primary administrative user name**. This is the identity you use to login to the administrative console or to wsadmin. When interoperating with a previous release, you must use the same server ID. In WebSphere Application Server Version 6.1, there are two choices for server ID. For previous releases, specify a server ID and password. However, if you choose to use the internal server ID, you must be able to override the generated value and to specify the server ID from a previous release.
 - d. Click either **Automatically generated server identity** or **Server identity that is stored in the user repository**.
 - e. If you select **Automatically generated server identity**, click **Authentication mechanisms and expiration** to change the identity to the one used by the previous release you are interoperating with. Scroll down to the Cross-cell single sign-on section and enter the identity in **Internal server ID**.
 - f. If you select **Server identity that is stored in the user repository**, enter the **Server user id** and the associated **Password**.
 - g. Fill out the rest of the user registry settings and then click **OK**.
2. Configure the LTPA authentication mechanism. Automatic generation of the LTPA keys should be disabled. If not, keys used by a previous release are lost. Export the current LTPA keys from WebSphere Application Server Version 6.1 and import them into the previous release.
 - a. In the administrative console select **Security > Secure administration, applications, and infrastructure**.
 - b. Click **Authentication mechanisms and expiration**.
 - c. Click the **Key set groups** link, then click the key set group that displays in the Key set groups panel.
 - d. Clear the **Automatically generate keys** check box.
 - e. Click **OK**, then click **Authentication mechanisms and expiration** in the path at the top of the Key set groups panel.
 - f. Scroll down to the Cross-cell single sign-on section, and enter a password to use for encrypting the LTPA keys when adding them to the file.
 - g. Enter the password again to confirm the password.
 - h. Enter the **Fully qualified key file name** that contains the exported keys.
 - i. Click **Export keys**.
 - j. Follow the instructions provided in the previous release to import the exported LTPA keys into that configuration.
3. If you are using the default SSL configuration, extract all of the signer certificates from the WebSphere Application Server Version 6.1 common trust store. Otherwise, extract signers where necessary to import them into the previous release.
 - a. In the administrative console, click **Security > SSL certificate and key management**.
 - b. Click **Key stores and certificates**.
 - c. Click **CellDefaultTrustStore**.
 - d. Click **Signer certificates**.
 - e. Select one signer and click **Extract**.

- f. Enter a unique path and filename for the signer (for example, c:\temp\signer1.arm).
 - g. Click **OK**. Repeat for all of the signers in the trust store.
 - h. Check other trust stores for other signers that might need to be shared with the other server. Repeat steps e through h to extract the other signers.
4. Add the exported signers to DummyServerTrustFile.jks and DummyClientTrustFile.jks in the /etc directory of the back-level product version. If the previous release is not using the dummy certificate, the signer certificate(s) from the previous release must be extracted and added into the WebSphere Application Server Version 6.1 release to enable SSL connectivity in both directions.
 - a. Open the key management utility, iKeyman, for that product version.
 - b. Start ikeyman.bat or ikeyman.sh from the \${USER_INSTALL_ROOT}/bin directory.
 - c. Select **Key Database File > Open**.
 - d. Open \${USER_INSTALL_ROOT}/etc/DummyServerTrustFile.jks.
 - e. Enter WebAS for the password.
 - f. Select **Add** and enter one of the files extracted in step 2. Continue until you have added all of the signers.
 - g. Repeat steps c through f for the DummyClientTrustFile.jks file.
 5. Verify that the application uses the correct Java Naming and Directory Interface (JNDI) name and naming bootstrap port for performing a naming lookup.
 6. Stop and restart all of the servers.

Interoperating with a C++ common object request broker architecture client

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 a client certificate to achieve mutual authentication on WebSphere Application Server applications.

V6.0.x Interoperability can be achieved between C++ CORBA clients and WebSphere Application Server using the Common Secure Interoperability Version 2 (CSIv2) or by using the z/OS Secure Authentication Services (z/SAS) protocols. CSIv2 should be used unless interoperability is required with a WebSphere Application Server Version 4.

Important: z/SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

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 a client certificate to achieve mutual authentication on WebSphere Application Server 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 presented in the following list in the file:

C++ security setting	Description
client_protocol_password	Specifies the password for the user ID.
client_protocol_user	Specifies the user ID to authenticate at the target server.
security_sslKeyring	Specifies the name of the RACF keyring for the client to use. The keyring must be defined under the user ID that is issuing the command to run the client.

- Point to the environment file using the fully qualified path name through the WAS_CONFIG_FILE environment variable. For example, in the test.sh test shell script, export:


```
/WebSphere/V6R0M0/DeploymentManager/profiles/default/config/cells  
/PLEX1Network/nodes/PLEX1Manager/servers/dmgr
```

Some of the environment file terms are explained below:

default

profile name

PLEX1Network

cell name

PLEX1Manager

node name

dmgr server name

Migrating custom user registries

If you built your own custom user registry, consider the migration items listed below. If you have a custom user registry that was provided by a Security Solution Provider, you must contact that provider to ensure that you have the correct version of their custom user registry to support WebSphere Application Server.

In WebSphere Application Server, 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.

You cannot use other WebSphere Application Server components, for example, data sources, to initialize the custom registry because other components, like the containers, are initialized after security and are not available during the registry initialization. A custom registry implementation is a pure custom implementation, independent of other WebSphere Application Server components.

The getCallerPrincipal enterprise bean method and the getUserPrincipal and getRemoteUser servlet methods return the security name instead of the display name. For more information, see the API documentation.

If the migration tool is used to migrate the WebSphere Application Server Version 5 configuration to WebSphere Application Server Version 6.0.x and later, this migration does not change your existing code. Because the WebSphere Application Server Version 5 custom registry works in WebSphere Application Server Version 6.0.x and later without any changes to the implementation, except when using data sources, you can use the Version 5-based custom registry after the migration without modifying the code.

Copy the files to the cell and to each of the node class paths.

In WebSphere Application Server Version 6.0.x and later, a case-insensitive authorization can occur when using an enabled custom user registry.

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

Before proceeding, look at the UserRegistry interface. See the article in the information center on developing standalone custom registries for a description of each of these methods in detail.

The following steps go through all the changes that are required to move your WebSphere Application Server Version 4.x custom user registry that implemented the old com.ibm.websphere.security.CustomRegistry interface to the com.ibm.websphere.security.UserRegistry interface.

Note: The sample implementation file is used as an **example** when describing the following steps.

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

```
public class FileRegistrySample implements CustomRegistry
```

to:

```
public class FileRegistrySample implements UserRegistry
```

2. Create the java.rmi.RemoteException exception 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 interested only in the first certificate, take the first certificate in the chain before processing. In WebSphere Application Server Version 6.0.x and later, 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.

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 that is obtained from the user 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 exception. Change the following code:

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

to:

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

In WebSphere Application Server Version 6.0.x and later, this method is not called directly by the WebSphere Application Server security component. However, other components of WebSphere Application Server, like the WebSphere Business Integration Server Foundation process choreographer, use this method when staff assignments are modeled using groups. Because this implementation is supported in WebSphere Application Server Version 6.0.x and later, 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 code:

```
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 that is obtained from the user registry whose number is limited to the value of the limit parameter. 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 are split onto two lines for illustrative purposes only.

11. To build the WebSphere Application Server Version 6.0.x and later implementation, make sure you have the sas.jar and the wssec.jar files in your class path.

To set the files in your class path, use the following code as a sample and substitute your environment values for the variables that are used in the example:

```
%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 WebSphere Application Server Version 5 custom registry (CustomRegistry) in WebSphere Application Server Version 6.0.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.

Make sure that you copy these classes to the cell and all the nodes. Without the classes in each of the node class paths, the nodes and the application servers in those nodes cannot start when security is on.

13. Use the administrative console to set up the custom registry.

Follow the instructions in the article in the information center on configuring standalone custom registries to set up the custom registry, including the **Ignore case for authorization** option. Make sure that you add the WAS_UseDisplayName properties if required.

WebSphere Application Server Version 4.x based custom user registry that implemented the old com.ibm.websphere.security.CustomRegistry interface is migrated to the com.ibm.websphere.security.UserRegistry interface.

If you are enabling security, see the article in the information center on enabling security to complete the remaining steps. When completed, save the configuration and restart all the servers. Try accessing some Java 2 Platform, Enterprise Edition (J2EE) resources to verify that the custom registry migration is successful.

Migrating trust association interceptors

Use this topic to manually migrate trust associations.

Note: Data sources are not supported for use within a Trust Association Interceptor (TAI). Data sources are intended for use within J2EE applications and designed to operate within the EJB and Web containers. Trust Association Interceptors do not run within a container, and while data sources may function in the TAI environment, they are untested and not guaranteed to function properly.

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 `com.ibm.websphere.security.webseal.ignoreProxy` property is 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 that are 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
```

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

The previous VIA header information was split onto two lines for illustrative purposes only.

For more information about the `com.ibm.websphere.security.webseal.ignoreProxy` property, see the article in the information center on configuring single signon using trust association interceptor ++.

Migrating product-provided trust association interceptors

The properties that are located in the `webseal.properties` and `trustedserver.properties` files are not migrated from previous versions of WebSphere Application Server. You must migrate the appropriate properties to WebSphere Application Server Version 6.0.x using the trust association panels in the administrative console. For more information, see the article in the information center on configuring trust association interceptors.

Changes to the custom trust association interceptors

If the custom interceptor extends the `com.ibm.websphere.security.WebSphereBaseTrustAssociationInterceptor` property, 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 that the interceptor is 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.initStatus` property in the trust association interceptor custom properties. Set the property to the value that indicates 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 that is required to initialize the interceptor. All of the properties set for an interceptor are 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 Zero (0) implies that the interceptor initialization is successful. Any other value implies that the initialization is not successful and the interceptor is not used.

The `init(String)` method still works if you want to use it instead of implementing the `init(Properties)` method. The only requirement is that you enter the file name containing the custom trust association properties 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 `myTAI.properties` file is located in the `app_server_root/properties` directory, set the following properties:

- `com.ibm.websphere.security.trustassociation.types = myTAItype`
- `com.ibm.websphere.security.trustassociation.myTAItype.config = app_server_root/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=  
app_server_root/properties/myTAI.properties
```

The previous line of code is split into two lines for illustrative purposes only. Type as one continuous line.

In a Network Deployment installation, where the location of the file name can vary for different nodes, use the variable `install_root` to refer to the WebSphere Application Server installation directory.

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

Migrating custom trust association interceptors

The trust associations from previous versions of WebSphere Application Server are not automatically migrated to WebSphere Application Server Version 6.0.x and later. You 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 code:

```
%WAS_HOME%/java/bin/javac -classpath %WAS_HOME%/lib/wssec.jar;  
%WAS_HOME%/lib/j2ee.jar your_implementation_file.java
```

The previous line of code is broken into two lines for illustrative purposes only. Type the code as one continuous line.

2. Copy the custom trust association interceptor class files to a location in your product class path. Copy these class files into the %WAS_HOME%/lib/ext directory.
3. Start WebSphere Application Server.
4. Enable security to use the trust association interceptor. The properties that are located in your custom trust association properties file and in the `trustedserver.properties` file are not migrated from previous versions of WebSphere Application Server. You must migrate the appropriate properties to WebSphere Application Server Version 6.0.x and later using the trust association panels in the administrative console.

For more information, see the article in the information center on configuring trust association interceptors.

Migrating Common Object Request Broker Architecture programmatic login to Java Authentication and Authorization Service (CORBA and JAAS)

Use this topic as an example of how to perform programmatic login using the CORBA-based programmatic login APIs.

Common Object Request Broker Architecture (CORBA) application programming interfaces (API) are not supported in the WebSphere Application Server for z/OS environment. If you have an application that you are porting from another WebSphere Application Server product to WebSphere Application Server for z/OS you must be aware that the security APIs that are deprecated in Version 6.0.x. If you want to use these applications on WebSphere Application Server for z/OS Version 6.0.x, you must migrate to Java Authentication and Authorization Service (JAAS).

The following list includes the deprecated CORBA programmatic login APIs.

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

The APIs that are provided in WebSphere Application Server are a combination of standard JAAS APIs and a product implementation of standard JAAS interfaces.

The supported APIs that are provided in WebSphere Application Server for z/OS are a combination of standard JAAS APIs and a product implementation of standard JAAS interfaces with some minor extension.

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` handler when the application pushes basic authentication data (user ID, password, and security realm) or token data to product login modules. This API is recommended for server-side login.

com.ibm.websphere.security.auth.callback.WSStdinCallbackHandlerImpl

Provides a stdin login prompt CallbackHandler handler 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 a LoginModules implementation for client and server-side login. Refer to the *Securing applications and their environment* 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`

An application must invoke the `WSSubject.doAs` method for J2EE resources access using the subject that is generated by an explicit invocation of a WebSphere Application Server login module.

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 Application Server LoginModules interface.

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 *Securing applications and their environment* 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 date 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 WSPincipal.
java.util.Set ps =
s.getPrincipals(com.ibm.websphere.security.auth.WSPincipal.class);
java.util.Iterator it = ps.iterator();
while (it.hasNext()) {
com.ibm.websphere.security.auth.WSPincipal p =
(com.ibm.websphere.security.auth.WSPincipal) 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

Use this topic to allow migration in an application that uses form-based login and servlet filters without the use of the CustomLoginServlet class.

The CustomLoginServlet class is deprecated in WebSphere Application Server 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 and 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 that is 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 JavaServer Pages (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 that contains the user name and the j_password parameter that contains 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.
Refer to the *Securing applications and their environment* PDF for details.
2. Configure the form login page and the error page for the application as described in .
Refer to the *Securing applications and their environment* PDF for details.
3. Develop servlet filters if additional processing is required before and after form login authentication.
Refer to the *Securing applications and their environment* PDF for details.
4. Configure the servlet filters that are 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.
For more information, see the *Securing applications and their environment* PDF.

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

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

Migrating Java 2 security policy

Use this topic for guidance pertaining to migrating Java 2 security policy.

Previous WebSphere Application Server releases

WebSphere Application Server uses the Java 2 security manager in the server runtime to prevent enterprise applications from calling the System.exit and the System.setSecurityManager methods. These two Java application programming interfaces (API) 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 not a beneficial operation for an application server.

To support Java 2 security properly, all the server runtime 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 that are 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 that are required by the server runtime. This situation is due to the design and algorithm that is used by Java 2 security to enforce permission checks. Refer to the Java 2 security check permission algorithm.

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

- `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 runtime is granted these permissions. All the other permission checks are not enforced.

Only two permissions are supported:

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

However, not all the product server runtime 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 WebSphere Application Server, which means that all permissions are enforced. The default Java 2 security policy for an enterprise application is the recommended permission set defined by the Java 2 Platform, Enterprise Edition (J2EE) Version 1.4 specification. Refer to the *profile_root/config/cells/cell_name/nodes/node_name/app.policy* file for the default Java 2 security policy that is granted to enterprise applications. This policy is a much more stringent compared to previous releases.

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

Note: The default Java 2 security policy for enterprise applications is much more stringent and all the permissions are enforced in WebSphere Application Server Version 6.0.x and later. The security policy might fail because the application code does not have the necessary permissions granted where system resources, such as file I/O, can be programmatically accessed and are now subject to the permission checking.

In application code, do not use the `setSecurityManager` permission to set a security manager. When an application uses the `setSecurityManager` permission, there is a conflict with the internal security manager within WebSphere Application Server. If you must set a security manager in an application for RMI purposes, you also must enable the **Use Java 2 security to restrict application access to local resources** option on the Secure administration, applications, and infrastructure page within the WebSphere Application Server administrative console. WebSphere Application Server then registers a security manager. The application code can verify that this security manager is registered by using `System.getSecurityManager()` application programming interface (API).

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). This system property contains both system permissions (permissions granted to the Java virtual machine (JVM) and the product server runtime) and enterprise application permissions. Migrate the Java 2 security policy of the

enterprise application to WebSphere Application Server Version 6.0.x. 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 system property 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 WebSphere Application Server (action might be required). This system property is deprecated; superseded by the `${user.install.root}` and `${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

No easy way exists to migrate the Java policy file to WebSphere Application Server Version 6.0.x and later automatically because of 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 code base is used instead of an absolute code base. This process has many advantages. Grant the permissions that are defined in the `was.policy` to the specific enterprise application only, while permissions in the `app.policy` file apply to all the enterprise applications that run on the node where the `app.policy` file belongs.

Refer to the *Securing applications and their environment* 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 for the `app1.ear` enterprise application and the system permissions, which are permissions that are granted to the Java virtual machine (JVM) and the product server runtime.

The default location for the Java 2 security policy file is `profile_root/properties/java.policy`. Default permissions are omitted for clarity:

```
// For product Samples
grant codeBase "file:${app_server_root}/installedApps/app1.ear/-" {
    permission java.security.SecurityPermission "printIdentity";
    permission java.io.FilePermission "${app_server_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 the file is not present, or update the `was.policy` file for migrated applications in the configuration repository 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 presented on two lines for illustrative purposes only.

The was.policy file is located in the *profile_root/config/cells/cell_name/applications/app.ear/deployments/app/META-INF/* directory.

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 *Securing applications and their environment* 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 `${user.install.root}/config/cells/cell_name/nodes/node_name/app.policy` file. This validation requires code review, code inspection, application documentation review, and sandbox testing of migrated enterprise applications with Java 2 security enabled in a preproduction 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 preproduction testing of the migrated enterprise application with Java 2 security enabled. Enable trace for the WebSphere Application Server Java 2 security manager in a preproduction testing environment with the following trace string: `com.ibm.ws.security.core.SecurityManager=all=enabled`. This trace function can be helpful in debugging the `AccessControlException` exception that is created 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 that are granted to the classes on the call stack when the exception is created.

For more information, see the *Securing applications and their environment* PDF.

Note: Because the Java 2 security policy is much more stringent compared with previous releases, the administrator or deployer must 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.

Enabling embedded Tivoli Access Manager

Embedded Tivoli Access Manager is not enabled by default, and you need to configure it for use.

Enabling Tivoli Access Manager security within WebSphere Application Server requires:

- A supported Lightweight Directory Access Protocol (LDAP) installed somewhere on your network. This user registry contains the user and group information for both Tivoli Access Manager and WebSphere Application Server.
- Tivoli Access Manager server exists and is configured to use the user registry. For details on the installation and configuration of Tivoli Access Manager, refer to the IBM Tivoli Access Manager for e-business information center.

Note: WebSphere Application Server contains an embedded client for Tivoli Access Manager. To use Tivoli Access Manager, you must also configure the Tivoli Access Manager server.

However, the server version must be the same version or later as the client version. For information on the supported version of Tivoli Access Manager, see *WebSphere Application Server - Supported Prerequisites*.

Tivoli Access Manager server is bundled with WebSphere Application Server.

- WebSphere Application Server is installed either in a single server model or as WebSphere Application Server Network Deployment.
- When administrative security is configured with a Federal Information Processing Standard (FIPS) provider, the Tivoli Access Manager server must be configured for FIPS as well

Complete the following steps to enable embedded Tivoli Access Manager security:

1. Create the security administrative user.
For more information, see the *Securing applications and their environment* PDF.
2. Configure the Java Authorization Contract for Containers (JACC) provider for Tivoli Access Manager .
For more information, see the *Securing applications and their environment* PDF.
3. Enable WebSphere Application Server security. When you are using Tivoli Access Manager you must configure LDAP as the user registry.
For more information, see the *Securing applications and their environment* PDF.
4. Enable the JACC provider for Tivoli Access Manager.
For more information, see the *Securing applications and their environment* PDF.

Propagating security policy of installed applications to a JACC provider using wsadmin scripting

It is possible that you have applications installed prior to enabling the Java Authorization Contract for Containers (JACC)-based authorization. You can start with default authorization and then move to an external provider-based authorization using JACC later.

Also, during application installation or modification you might have had problems propagating the security policy information to the JACC provider. For example, network problems might occur, the JACC provider might not be available, and so on. For these cases, the security policy of the previously installed applications does not exist in the JACC provider to make the access decisions. One choice is to reinstall the applications involved. However, you can avoid reinstalling by using the wsadmin scripting tool. Use this tool to propagate information to the JACC provider independent of the application installation process. The tool eliminates the need for reinstalling the applications.

The tool uses the SecurityAdmin MBean to propagate the policy information in the deployment descriptor of any installed application to the JACC provider. You can invoke this tool using wsadmin at the base application server for base and deployment manager level for Network Deployment. Note that the SecurityAdmin MBean is available only when the server is running.

Use `propagatePolicyToJACCProvider(String appNames)` to propagate the policy information in the deployment descriptor of the enterprise archive (EAR) files to the JACC provider. If the `RoleConfigurationFactory` and the `RoleConfiguration` interfaces are implemented by the JACC provider, the authorization table information in the binding file of the EAR files is also propagated to the provider. See the *Securing applications and their environment* PDF for more information about these interfaces.

The `appNames` String contains the list of application names, delimited by a colon (:), whose policy information must be stored in the provider. If a null value is passed, the policy information of the deployed applications is propagated to the provider.

Also, be aware of the following items:

- Before migrating applications to the Tivoli Access Manager JACC provider, create or import the users and groups that are in the applications to Tivoli Access Manager.
 - Depending on the application or the number of applications that are propagated, you might have to increase the request time-out period either in the `soap.client.props` file in the directory `profile_root/properties` (if using SOAP) or in the `sas.client.props` file (if using RMI) for the command to complete. You can set the request time-out value to 0 to avoid the timeout problem, and change it back to the original value after the command is run.
1. Configure your JACC provider in WebSphere Application Server.
See the *Securing applications and their environment* PDF for more information.
 2. Restart the server.
 3. Enter the following commands:

```

// use the SecurityAdmin MBean at the Deployment Manager or the unmanaged base application
// server connect to the appropriate process (Deployment Manager or base application server)
wsadmin -user serverID -password serverPWD
// To get the SecurityAdmin MBean for Deployment Manager
wsadmin> set secadm [$AdminControl queryNames type=SecurityAdmin,process=dmgr,*]
// or to get the SecurityAdmin MBean for a unmanaged base application server (replace the
// process name to match your configuration)
wsadmin> set secadm [$AdminControl queryNames type=SecurityAdmin,process=server1,*]
// to propagate specific applications security policy information
wsadmin>set appNames [list app1:app2]
// or to propagate all applications installed
wsadmin>set appNames [list null]

// Run the command to propagate
wsadmin>$AdminControl invoke $secadm propagatePolicyToJACCProvider $appNames

```

Naming and directory

JNDI interoperability considerations

You must take extra steps to enable your programs to interoperate with non-WebSphere Application Server JNDI clients and to bind resources from MQSeries to a name space.

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 is 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.

Single server

Following is a URL that has the bootstrap host **myHost**, the port **2809**, and the enterprise bean installed in the server **server1** in node **node1** and bound in that server under the name **myEJB**:

```

initialContext.lookup(
    "corbaname:iiop:myHost:2809#cell/nodes/node1/servers/server1/myEJB");

```

Server cluster

Following is a URL that has the bootstrap host **myHost**, the port **2809**, and the enterprise bean installed in a server cluster named **myCluster** and bound in that cluster under the name **myEJB**:

```

initialContext.lookup(
    "corbaname:iiop:myHost:2809#cell/clusters/myCluster/myEJB");

```

The lookup works with any name server bootstrap host and port configured in the same cell.

The lookup also works if the bootstrap host and port belong to a member of the cluster itself. To avoid a single point of failure, the bootstrap server host and port for each cluster member can 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.

Application profiling

Running Version 5 Application Profiles on Version 6

Java 2 platform, Enterprise Edition (J2EE) 1.3 applications created using WebSphere Application Server Version 5.x have an application profile configuration formatted for Version 5.x. Although you can use applications with an application profile configuration from Version 5.x on WebSphere Application Server Version 6.x, you must change a setting. Also, there are several implications to using version 5.x application profiles on version 6.x.

The WebSphere Application Server 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. See the *Developing and deploying applications* PDF for more information.

- Set the Application Profile service on your server to enable the Application Profiling 5.x Compatibility Mode as the default.

See the *Developing and deploying applications* PDF.

This setting is necessary to support Java 2 platform, Enterprise Edition (J2EE) 1.3 applications with an application profile configuration from WebSphere Application Server Version 5.x. 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, but no J2EE 1.3 applications were installed at server start up. Therefore, the server run time automatically ignores the 5.x compatibility in order to avoid performance costs associated with it.
2. you try to install and start a J2EE 1.3 application with an application profile configured in Version 5.x, but the 5.x compatibility mode is turned off.

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, or restart the server after installing a J2EE 1.3 application with the application profile configured in Version 5.x.

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

See the *Developing and deploying applications* PDF.

- Migrate any application you have configured with application profiling in Version 5.

Application profiles migration requires you to re-configure your applications in the Application Server Toolkit (AST). See the *Developing and deploying applications* PDF for more information.

- Rework the usage of the TaskNameManager API if it is used in your applications. The TaskNameManager API is not supported in container-managed transaction beans, and the `setTaskName()` method must be called before beginning a new unit of work.

See the *Developing and deploying applications* PDF for more information.

Migrating Version 5 Application Profiles to Version 6

You can migrate an application with a WebSphere Application Server Version 5.x application profile to a version 6 application profile. You might want to migrate because the application profiling function works under a different concept in version 6 and a major difference exists between Version 5.x and Version 6 in the scoping of a task name.

For the implications of using Java 2 platform, Enterprise Edition (J2EE) 1.3 applications with an application profile configuration from WebSphere Application Server Version 5.x on Version 6, see “Running Version 5 Application Profiles on Version 6” on page 73.

In Version 5.x, a task name could be assigned to a bean, web, or client method. When a method was invoked that had a task name configured on it, either that task name became the active task name (known as ‘run as specified’) or the caller’s task name, should one exist, became the active task name (known as ‘run as caller’). In Version 6, the application profiling function works under the unit of work (UOW) concept. UOW in this case means either a transaction or an ActivitySession. This means that the task name on a method is used only when a UOW is begun, because of that method being invoked. This gives it a more predictable data access pattern based on the active unit of work. To be more specific, this approach ensures that a bean type with only one configured access intent is loaded within a UOW, because a bean is configured with only one access intent within an application profile. This configured access intent for a bean type is determined at assembly time and is enforced by the Application Profile service. In Version 5.x, a bean type could be loaded in different access intents and cause deadlock problems within a UOW.

A major difference between Version 5.x and Version 6 is with the scoping of a task name. In Version 5.x, a task is not necessarily associated with a UOW. In Version 6, a task name is always associated with a UOW and that task name does not change for the duration of that UOW. When a UOW associated with a method is begun because of that method being invoked, if a task name is associated with the method then that task name is used to name the UOW. A task assigned to a UOW is considered a named UOW. If a task name is not associated with the method that began the UOW, then a default access intent is used and the UOW is unnamed. A UOW can only be named when the UOW is begun. Any task names associated with a method are ignored if that method does not begin a UOW (either container managed or component managed).

To migrate a Version 5.x application to use version 6 application profile functionality do the following:

1. Configure a UOW on a method.

In Version 5.x it wasn’t necessary to associate a task with a UOW. In Version 6, a task can only be associated with a UOW when that UOW is begun. Therefore, the first step to convert a Version 5.x application to Version 6 is to define a UOW on any method that has a task name (and eventually an Application Profile) associated with it. A UOW can either be container managed or component managed and can either be a transaction or an ActivitySession.

2. Associate a method level task name with the same methods used to define the UOW you just created.

When a method is called that causes a UOW to begin, if a task name is associated with the same method then that task name becomes the active task name and is associated with the UOW. Therefore, a task name needs to be defined on a method that participates in a UOW and that needs an Application Profile associated with it. A task can be associated with a method either declaratively through an application's deployment descriptor or programmatically through the TaskNameManager API.

The *Developing and deploying applications* PDF describes how to configure task names either declaratively or programmatically.

3. Reassign 'run as caller' tasks. In Version 5.x, you could define a container task to be either 'run as caller' or 'run as specified'. In the 'run as caller' case, if the configured methods are invoked with an associated task, the method is executed with the imported (caller's) task. If a method is invoked by a request that is not associated with a task, the method continues to execute without a task. In the 'run as specified' case, the configured methods are never invoked with an imported (caller's) task. Instead, the method executes as the specified task name.

In Version 6, these two "run as" options have explicitly been removed yet are still implied. In other words, when a user defines a method with a task name, the user is specifying a task name to be used when the method is invoked. This is similar to 'run as specified' in Version 5.x. However, it is **not exactly** 'run as specified' because in Version 6 a task name can only be associated with a UOW when that UOW is begun. In other words, even if a method has a task name associated with it in Version 6, that task name is ignored if there is already a UOW, or if the method does not cause a UOW to begin. In Version 5.x, a 'run as specified' task defined on a method always became the active task name when the method was invoked.

The 'run as caller' option of Version 5.x is also implicitly available in Version 6, although not explicitly stated. When a method (callee) is invoked by another method (caller) that has a UOW associated with it, the callee is associated with the caller's UOW if the callee does not begin a new UOW. In other words, the callee runs with the caller's task. The name of the UOW is the task name associated with the callee, even if the callee has its own task name specified. Keep in mind that the UOW could be unnamed, in which case no task name is associated with the caller or callee.

4. **Optional:** Remove method level access intent. In Version 6.0, method level access intent has been deprecated. It is suggested that you remove method level access intent from your applications. In its place assign task names to a method and then assign the task name to an Application Profile with the necessary access intent.

Application profiling interoperability

Using application profiling with 5.x compatibility mode or in a clustered environment with mixed WebSphere Application Server product versions and mixed platforms can affect its behavior in different ways.

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 or later 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.

Similarly, application profiles that you create using the latest version of WebSphere Application Server are not compatible with Version 5 or earlier versions. Even applications configured with application profiles run on Version 6.x 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's 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.x 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.x.

If a clustered environment contains both Version 5.x and 6.x 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.x 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.

Asynchronous beans

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 the topic, Chapter 5, “Migrating product configurations,” on page 87.

1. Install the Version 6.x product.
2. Migrate the previous release to the Version 6.1 product.
See “Overview of migration, coexistence, and interoperability” on page 77.

Chapter 4. Migrating and coexisting

Migrating involves collecting the configuration information from a previous release of a WebSphere Application Server product and merging it into a configuration for a new release. Coexisting involves 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.

See “Overview of migration, coexistence, and interoperability” and “Premigration considerations” on page 81.

The migration tools basically save the existing WebSphere configurations and user applications in a backup directory and then process the contents of this backup directory to migrate the configurations and your applications from previous WebSphere Application Server releases to the latest release.

If you have a previous version, you must plan whether to migrate the configuration and applications of the previous version to the new version. Migration does not uninstall the previous version. The earlier release is still functional. If you run the earlier release at the same time as the WebSphere Application Server for z/OS Version 6.1 installation, the two versions are *coexisting*. The migration articles describe migration from WebSphere Application Server for z/OS Version 5.x or 6.0.x to WebSphere Application Server for z/OS Version 6.1.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

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

For information on migrating to Version 6.1, see Chapter 5, “Migrating product configurations,” on page 87. For more information on coexistence among releases, see “Coexistence support” on page 141.

Overview of migration, coexistence, and interoperability

The goal of migration is to reconstruct your earlier version of WebSphere Application Server in a nearly identical Version 6.1 environment. One goal of coexistence is to create a mixed-version environment that is not in conflict and allows the nodes of all versions to start and run at the same time; another goal is to create an environment that facilitates rollback and allows one or the other version to run at one time. Interoperating is exchanging data between two coexisting product installations or between products on different systems.

Can I simply point to the new WebSphere Application Server for z/OS Version 6.1 data sets and restart my servers?

No. WebSphere Application Server for z/OS Version 6.1 requires that you migrate your Version 5.x or 6.0.x configuration up to the Version 6.1 level.

Be aware of the following issues when migrating to Version 6.1:

- Any variables that belong to applications or products other than WebSphere Application Server will not be migrated but will be brought over as is. Therefore, check any other product upgrades before migrating to ensure that all of these variables will still be accurate after migration.
- Before performing migration from Version 5.x or 6.0.x to Version 6.1, make sure that you do not have any region constraints (such as IEFUSI limits) in place. These can cause unpredictable JVM errors.

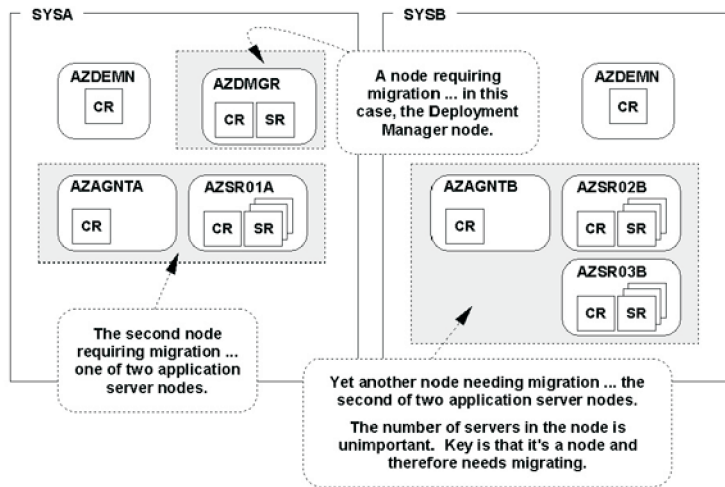
What is the basic migration process?

1. Install the SMP/E code for WebSphere Application Server for z/OS Version 6.1.
2. Use the Customization Dialog walk-throughs to create the migration utilities that you need to perform the migration.
3. Run these jobs.

A new Version 6.1 configuration is created—separate from your existing Version 5.x or 6.0.x configuration—that is based on the Version 5.x or 6.0.x configuration information.

Is migration a node-by-node activity?

Yes. The process of migrating the configuration involves running the supplied utilities against each node in your configuration.



With a stand-alone application server you have only one node, but that node needs to be migrated. The steps are essentially the same as you would perform for any other node, except you do not have to have a deployment manager up and running. See “Checklist of activities for migrating a stand-alone application server” on page 100 for a checklist of activities for migrating a stand-alone application server node.

What do the migration utilities do?

The migration utilities serve the following purposes:

Utility	Purpose
BBOWMG1B (stand-alone application server migrations)	Enables all servers on the node being migrated to be configured to start in PRR processing mode. After this job completes, you must start all servers on the node being migrated and wait for them to terminate. PRR processing mode resolves any outstanding transactions, clears the transaction logs, and terminates the server. This job is not needed for a deployment manager migration, and it is optional for configurations that do not use XA connectors.
BBOWMG1F (federated node migrations)	
	This job is required only if you are using XA adapters and you need to migrate the XA logs. Check your resource providers in the Version 5.x or 6.0.x administrative console by going to Resources > JDBC providers and checking to see if you have chosen any XA providers such as DB2, Cloudscape, and so on.

BBOWMG2B (stand-alone application server migrations) BBOWMG2F (federated node migrations)	Disables PRR mode and returns all servers to normal operating state. There is no need to start all servers after this job completes. This job is not needed for a deployment manager migration, and it is optional for configurations that do not use XA connectors. This job is required only if you are using XA adapters and you need to migrate the XA logs. Check your resource providers in the Version 5.x or 6.0.x administrative console by going to Resources > JDBC providers and checking to see if you have chosen any XA providers such as DB2, Cloudscape, and so on.
BBOMBHFS or BBOMBZFS (stand-alone application server migrations) BBOMDHFS or BBOMDZFS (deployment manager migrations) BBOMMHFS or BBOMMZFS (federated node migrations)	Optional: Creates a file system and mount point for the Version 6.1 configuration root, and mounts the file system. If you want to use an existing file system to contain the Version 6.1 configuration, you must manually create the mount point specified in the Customization Dialog and make sure that the file system is mounted rather than run this job. In either case, the configuration file system and mount point must be created and the file system must be mounted before proceeding with the migration.
BBOWMG3B (stand-alone application server migrations) BBOWMG3D (deployment manager migrations) BBOWMG3F (federated node migrations)	Performs the migration of the node from Version 5.x or 6.0.x to Version 6.1.
BBOMBBCP (stand-alone application server migrations) BBOMDCP (deployment manager migrations) BBOMMCP (federated node migrations)	Copies the generated JCL procedures to start the servers to the specified procedure library. If you choose to have your Version 6.1 configuration make use of different JCL start procedure names, this utility will update the new Version 6.1 configuration, substituting your new JCL names in place of the names that existed in your original Version 5.x or 6.0.x configuration.

Where should the migration jobs be run?

Run the jobs on the same system on which the node being migrated resides.

What happens when a node is migrated?

The migration utilities transform the contents of your present WebSphere Application Server Version 5.x or 6.0.x configuration file system and merge them into a new, separate Version 6.1 configuration file system.

Will my existing configuration be lost during migration?

During the migration, the original WebSphere Application Server Version 5.x or 6.0.x configuration tree is unaffected. If for some reason the migration fails before completing, your previous configuration still exists.

If my node has multiple application servers, will all of them be migrated?

Yes. The utility will detect all servers and migrate all, including the node agent. One invocation of the migration utilities against the node will take care of all the servers in the node.

Do the servers in a node have to be stopped to perform the migration?

Yes. In a multinode configuration it is possible to have the other nodes still running. But any node that you want to migrate must have its servers stopped.

Note: When an application server node that is part of a Network Deployment configuration is being migrated, the previously migrated Version 6.1 deployment manager for that cell must be up and running. This is because part of the migration involves the use of the **wsadmin** scripting function to synchronize the newly migrated application server node with the deployment manager. The deployment manager must be up to perform that synchronization.

Is it possible to have a cell operating with only some of the nodes migrated and others not?

Yes, that is possible. WebSphere Application Server Version 5.1 can coexist with Version 6.1 in the same cell and on the same LPAR. When migrating from Version 5.0.x, however, you need to migrate the deployment manager node and other application server nodes on that same MVS image one right after the other — or essentially at the same time. Version 5.0.x and Version 6.1 nodes cannot exist in the same cell on the same LPAR.

Can my newly-migrated WebSphere Application Server for z/OS Version 6.1 deployment manager still "talk" to Version 5.x or 6.0.x nodes?

Yes. A deployment manager migrated to the Version 6.1 level of code can manage a Version 5.x or 6.0.x node. Changes made through the administrative console will be applied to the node. There are a few things to keep in mind:

- When a deployment manager is migrated to Version 6.1, a new Version 6.1 "master configuration" is created. The Version 5.x or 6.0.x master configuration still exists. But when the Version 6.1 deployment manager makes changes to the configuration, it makes them to the new Version 6.1 master configuration. So while it is possible to use the Version 5.x or 6.0.x code, any changes made in Version 6.1 will not be seen when the older code is restored.
- A Version 5.x or 6.0.x deployment manager has no ability to manage a Version 6.1 node.
- A WebSphere Application Server Version 6.1 deployment cell can contain mixed releases of Version 5.x or 6.0.x nodes, but there is no mixed-node management support for Version 6.0.1.x.
The Version 6.1 migration tools still migrate these nodes during deployment-manager migration, but they issue a warning message that the nodes cannot be managed by the Version 6.1 deployment manager. You can then do one of the following based on your needs.
 - Upgrade all Version 6.0.1.x nodes to at least Version 6.0.2. This will allow them to be administered by a Version 6.1 deployment manager.
 - Immediately migrate these nodes to Version 6.1.

Is there a sequence to performing a multinode migration?

Yes, there is.

1. Always migrate the deployment manager first.
2. WebSphere Application Server for z/OS Version 5.0.x application servers that reside on the same node as the deployment manager must be migrated next. Version 5.1 and Version 6.1 nodes can coexist in the same cell and on the same LPAR, so any Version 5.1 nodes do not need to be migrated immediately. See "Coexistence support" on page 141 for more information on coexistence.
3. Application server nodes on the same system as the deployment manager or on other MVS images can then be migrated.

Is it possible to have cells at WebSphere Application Server for z/OS Version 6.1 coexist with other cells at Version 5.x or 6.0.x?

Yes, it is. This is true for a sysplex or any given MVS image. There are some restrictions, most of which have been present in past versions as well:

- No two cells can have the same cell short name.
- Version 5.0.x cannot coexist with Version 6.1 in the same cell on the same LPAR. If you have multiple Version 5.0.x nodes on the same LPAR, all nodes must be migrated to Version 6.1 at the same time.
- Only one version of the code can exist in LPA/LNKLST; the rest must be included in STEPLIB.
- There are other things you must consider for separate cells, regardless of whether they are at different versions of the code; for example, you must have a separate configuration file system mount point and separate JCL procedures.

Premigration considerations

Before you begin the process of migrating to WebSphere Application Server Version 6.1, there are some considerations of which you need to be aware.

- After you have installed WebSphere Application Server for z/OS Version 6.1, you might want to build a complete deployment cell configuration and make sure that it works properly before you attempt to migrate an existing cell or node.

This ensures that your system has all of the necessary prerequisites and supports the new level of WebSphere Application Server.

- WebSphere Application Server Version 6.1 does not provide support for migrating a WebSphere Application Server Version 5.x configuration that contains an internal JMS provider.

The JMS applications must be manually installed and reconfigured after migration to use the WebSphere Application Server Version 6.1 JMS provider.

- If you are migrating from WebSphere Application Server for z/OS Version 6.0.2.x to Version 6.1 and your configuration uses XA connections, you must have Version 6.0.2.9 or later before migrating.
- If you are currently running your WebSphere Application Server for z/OS Version 5.0.x node using STEPLIB, you must verify that *profile_root/bin/setupCmdLine.sh* contains the STEPLIB of the load libraries.

Some Version 5.0.x nodes might not contain the installation-generated STEPLIB statement. In these instances, you must add the STEPLIB manually to the "OS/390)" section of the *setupCMDLine.sh* file before running the migration utilities. Here is an example from a *setupCMDLine.sh* file in a properly configured system:

```
STEPLIB=BOSS.VICOM.W000020.SBBOLPA:$STEPLIB
STEPLIB=BOSS.VICOM.W000020.SBBOLD2:$STEPLIB
STEPLIB=BOSS.VICOM.W000020.SBBLOAD:$STEPLIB
export STEPLIB
```

- Migration support requires that both the source and target WebSphere Application Server for z/OS systems are on the same LPAR.

Therefore, you cannot migrate an existing configuration to a different z/OS LPAR. You also cannot migrate to or from a non-z/OS operating system using the WebSphere Application Server Version 6.1 migration utilities.

- Migrating cells that span Sysplex environments or operating systems should not present any unique migration issues. You migrate at the node level, and you use the tools provided based on the platform of the node that you are migrating.

Note: For information on setting up a mixed-platform cell, see the *WebSphere for z/OS -- Heterogeneous Cells* white paper that is available at <http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100644>.

- WebSphere Application Server for z/OS does not support the **WASPreUpgrade**, **WASPostUpgrade**, and **manageprofiles** command-line tools that are supported by the distributed and i5/OS versions of the product.

You must generate the migration jobs using the Customization Dialog and then submit them according to the generated instructions.

- Before you migrate to JDK 1.5 (introduced in WebSphere Application Server Version 6.1) from JDK 1.4 (introduced in Version 5.1) or JDK 1.3 (supported in Version 5.0.x) , review your applications for necessary changes based on the Sun Microsystems Java specification.

See “API and specification migration” on page 84,

- When migrating a cell with multiple nodes, the applications must remain at lowest JDK level until all nodes are migrated.
- The Web server plug-in configuration file (plugin-cfg.xml) generated after successful migration from Version 5.x to Version 6.x is topology centric—that is, it includes all the applications within a cell. You cannot manage this cell-wide plug-in configuration file from the administrative console until you have manually configured it.

See Chapter 6, “Migrating Web server configurations,” on page 135,

- The migration articles assume that WebSphere Application Server Version 6.1 is being installed in an environment where it must coexist with prior levels of WebSphere Application Server.

Consider the following items while planning to enable coexistence:

- Update prerequisites to the levels required by WebSphere Application Server Version 6.1.
Prior levels of WebSphere Application Server will continue to run at the higher prerequisite levels.
- Set up WebSphere Application Server for z/OS Version 6.1 to eliminate potential LPA conflicts with a prior Version 5.x or 6.0.x installation.
WebSphere Application Server Version 5.x, 6.0.x, and 6.1 require the placement of some code into LPA (SBBOLPA). In addition, additional product code (SBBOLoad) should be placed into LPA for performance reasons. Because of naming conflicts, however, more than one version of product code can not be in LPA at the same time.
- Review the ports that have been defined to ensure that the WebSphere Application Server Version 6.1 installation does not conflict.

In particular, note that the default daemon port definition for both versions is the same when installing to coexist with WebSphere Application Server Version 5.x or 6.0.x.

See Chapter 8, “Coexisting,” on page 141.

- Consider the following if you are planning to have any mixed-release cells:
 - You can upgrade a portion of the nodes in a cell to WebSphere Application Server Version 6.1 while leaving others at the older release level. This means that, for a period of time, you might be administering servers that are at the current release and servers that are running the newer release in the same cell.
Note that in this mixed-release environment, there might be some restrictions on what you can do with servers at the older release level. There might also be restrictions on creating clusters and cluster members. For details, see the information center.
 - WebSphere Application Server Version 5.0.x cannot coexist with Version 6.0.x or Version 6.1 in the same cell on the same system image (LPAR).
If you have multiple WebSphere Application Server Version 5.0.x nodes, including the deployment manager, on the same LPAR, all nodes must be migrated to WebSphere Application Server Version 6.1 at the same time. All nodes must be migrated sequentially before starting any WebSphere Application Server Version 6.1 node in the LPAR.
 - A WebSphere Application Server Version 6.1 deployment cell can contain mixed releases of Version 5.x or 6.0.x nodes, but there is no mixed-node management support for Version 6.0.1.x.
The Version 6.1 migration tools still migrate these nodes during deployment-manager migration, but they issue a warning message that the nodes cannot be managed by the Version 6.1 deployment manager. You can then do one of the following based on your needs.
 - Upgrade all Version 6.0.1.x nodes to at least Version 6.0.2. This will allow them to be administered by a Version 6.1 deployment manager.
 - Immediately migrate these nodes to Version 6.1.

- WebSphere Application Server Version 6.1 migration converts HTTP transports to channel-framework Web container transport chains.
- Include maintenance considerations when developing your configuration file system strategy.

If you configure your Network Deployment environment using the default value for the product file system path in the Customization Dialog, it will result in all the nodes pointing directly at the mount point of the product file system. This makes rolling maintenance in a nondisruptive manner almost impossible. If a cell is configured in this way, applying service to the product file system affects all the nodes at the same time; and if multiple cells are configured in this way, applying service to the product file system affects all the cells at the same time.

You might want to specify what is referred to as an "intermediate symbolic link" between each node's configuration file system and the actual mount point of the product file system. This strategy is described in the WebSphere Application Server for z/OS V5 - Planning for Test, Production and Maintenance white paper.

See the Washington Systems Center Sample WebSphere for z/OS ND Configuration white paper for more information about this issue and its relationship to applying maintenance. See the WebSphere for z/OS: Updating an Existing Configuration HFS to Use Intermediate Symbolic Links instructions for information on obtaining and using a utility that would allow you to update an existing configuration HFS to use intermediate symbolic links.

- The migration tools create a migration backup directory containing a backup copy of the configuration from the previous version. The space available for this directory should be at least the size of the previous version's configuration directory and applications plus the size of the batch-job output from the migration.

Normally, the batch-job output from the migration is very small unless you enable trace. The trace output size varies depending on the parts of migration for which you have enabled trace. The largest trace producer is the WASPostUpgrade phase of migration. You can typically see trace output of around 30 MB for this phase.

- WebSphere Application Server Version 6.1 does not support the DB2 for zOS Local JDBC Provider (RRS).

If you use the DB2 for zOS Local JDBC Provider (RRS) in the Version 5.x or 6.0.x configuration that you are going to migrate, you must change your configuration to use the DB2 Universal JDBC Driver Provider before or immediately after you migrate to Version 6.1. The Version 6.1 migration tools do not migrate the provider for you.

If you use the DB2 for zOS Local JDBC Provider (RRS) in the version to be migrated and do not change your configuration to use the DB2 Universal JDBC Driver Provider before migrating to Version 6.1, the following events will occur:

- When running the migration tools, you will receive the following message:
MIGR0442W: Not migrating DB2 for zOS Local JDBC Provider (RRS) jdbc provider.
Manually create a DB2 Universal Driver provider as a replacement. See DB2 documentation for further details.
- After migration, DB2 access will be broken and you will receive the following runtime message:
DSRA8213W: JDBC provider, DB2 for zOS Local JDBC Provider (RRS), is no longer supported by WebSphere Application Server. Applications should use DB2 Universal JDBC Driver Provider Type 2.

If you determine that you must change your configuration to use the DB2 Universal JDBC Driver Provider, you can do so by performing one of the following tasks:

- Before migrating to Version 6.1, change your Version 5.x or 6.0.x configuration to use the DB2 Universal JDBC Driver Provider.

If you do this, the Version 6.1 migration tools will handle the migration to the DB2 Universal JDBC Driver Provider and there will be no post-migration activity required.

Perform one of the following actions:

- Manually change your configuration to use the DB2 Universal JDBC Driver Provider.

Search the information center for your Version 5.x or 6.0.2 product to find information on configuring a DB2 Universal JDBC Driver Provider for WebSphere Application Server for z/OS.

- If you are migrating from Version 5.1.0 or later, use the provider-migration tool to migrate from the DB2 for zOS Local JDBC Provider (RRS) to the DB2 Universal JDBC Driver Provider.

This tool is a Jython script that migrates DB2 for zOS Local JDBC Providers (RRS) to DB2 Universal JDBC Driver Providers one node at a time. A white paper that accompanies the tool explains how to install and configure the DB2 Universal JDBC Driver before running the tool to migrate your configuration. The tool and white paper are available from the product Support site.

Note: If you are migrating directly from Version 5.0.2 to Version 6.1 and want to use this tool to migrate your providers, you must do so after migrating.

- After migrating to Version 6.1, perform one of the following actions:
 - Manually change your configuration to use the DB2 Universal JDBC Driver Provider.
 - Use the provider-migration tool to migrate from the DB2 for zOS Local JDBC Provider (RRS) to the DB2 Universal JDBC Driver Provider.

This tool is a Jython script that migrates DB2 for zOS Local JDBC Providers (RRS) to DB2 Universal JDBC Driver Providers one node at a time. A white paper that accompanies the tool explains how to install and configure the DB2 Universal JDBC Driver before running the tool to migrate your configuration. The tool and white paper are available from the product Support site.

- After you use the migration tools to migrate to WebSphere Application Server Version 6.1, you might need to do some things that are not done automatically by the migration tools.
 - Examine any Lightweight Third Party Authentication (LTPA) security settings that you might have used in WebSphere Application Server Version 5.x or 6.0.x, and make sure that Version 6.1 security is set appropriately.
 - 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 of 50.
 - Verify the results of the automatic Cloudscape database migration, and manually migrate any Cloudscape databases that are not automatically migrated by the tools.

See “Migrating Cloudscape databases” on page 130.

API and specification migration

Migrating application programming interfaces (APIs) and specifications involves moving to the current Java component level as well as to other technologies that WebSphere Application Server Version 6.1 supports.

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

In many cases, IBM provides additional features and customization options that extend the specification level even further. If your existing applications use IBM extensions from earlier product versions, it might be necessary for you to perform mandatory or optional migration to use the same kinds of extensions in Version 6.1.

The following table summarizes potential migration areas.

Functional area	Support in Version 5.0.x	Support in Version 5.1	Support in Version 6.0.x	Support in Version 6.1	Migration details
Enterprise JavaBeans (EJB)	EJB 2.0	EJB 2.0	EJB 2.1	EJB 2.1	Full support for EJB 2.0 and 2.1 is provided.

Functional area	Support in Version 5.0.x	Support in Version 5.1	Support in Version 6.0.x	Support in Version 6.1	Migration details
Java Connector Architecture (JCA)	JCA 1.0	JCA 1.0	JCA 1.5	JCA 1.5	Java 2 Connector support was completed in Version 6.0. Some changes might be necessary to take full advantage of this support.
Java Database Connectivity (JDBC) API	JDBC 2.0	JDBC 2.0	JDBC 3.0	JDBC 3.0	Many applications can run unchanged in Version 6.1 although some changes might be required or recommended.
JavaServer Pages (JSP)	JSP 1.2	JSP 1.2	JSP 2.0	JSP 2.0	For more information, see the Web applications article in the information center.
Security	IBM security	IBM security	IBM security	IBM security	Changes might be required due to J2EE security.
Java Servlets	Servlet 2.3	Servlet 2.3	Servlet 2.4	Servlet 2.4	Many Servlet 2.3 and 2.4 applications can run unchanged in Version 6.1 although changes might be required or recommended.
Sessions	IBM sessions	IBM sessions	IBM sessions	IBM sessions	Many applications can run unchanged in Version 6.1 although changes might be required or recommended.
Transactions	IBM transactions	IBM transactions	IBM transactions	IBM transactions	There was a change in the import statement in Version 6.0.x. Also, one datasource connection cannot be used across multiple user transactions.
Web services	Apache SOAP 2.3 in Version 5.0 and 5.0.1; Web Services for J2EE 1.1 in Version 5.0.2	Web Services for J2EE 1.1	Web Services for J2EE 1.1	Web Services for J2EE 1.1	The Apache SOAP Web services support provided in WebSphere Application Server Version 5.0 and 5.0.1 was deprecated in Version 5.0.2 and is not supported in Version 6.1. Applications that are using this SOAP implementation should migrate to Web Services for J2EE.
XML parser	XML4J 2.0.x	XML4J 4.0.6	XML4J 4.0.6	XML4J 4.2.2	Changes to move to the supported API XML4J Version 4.2.2 level are required.
					Recompilation is required to migrate a Version 5.x XML application to the Version 6.x level.
XML transformer	XSLT4J 2.5.4	XSLT4J 2.5.4	XSLT4J 2.5.4	XSLT4J 2.5.4	Changes are required to move to the supported XSLT4J 2.5.4 transformer levels.

Notes on the use of JDK 1.5:

- WebSphere Application Server began supporting JDK 1.5 in Version 6.1.
- In general, existing Version 5.x and 6.0.x application binaries that were developed using JDK 1.3 or 1.4 are highly compatible and usually do not require modifications to run. However, recompilation of the JDK 1.3 or 1.4 applications at the JDK 1.5 level might necessitate modifications of the source code to conform to incompatible changes present in JDK 1.5. As part of your migration planning, you should review the JDK compatibility restrictions that are documented by Sun Microsystems at Java 2 Platform Standard Edition 5.0 Compatibility with Previous Releases.
- A mixed cell containing Version 5.x or 6.0.x and Version 6.1 nodes requires that all application binaries remain at the lowest JDK level used. Although you can successfully migrate Version 5.x or 6.0.x applications to Version 6.1, this is only meant to be a temporary state as you transition to Version 6.1. After you begin migration to Version 6.1, you should plan to complete the migration of the entire cell, update your tooling to Version 6.1, and update your applications to conform to

JDK 1.5 requirements. This should be done before any further application changes. After you have completely migrated your cell to Version 6.1, you should upgrade your application binaries to the JDK 1.5 level the next time you make application modifications that require recompiling. This might require source code changes to your application to conform to the JDK 1.5 API changes as documented by Sun Microsystems.

Programming model extension migration

This product edition contains several programming model extensions (PMEs) that previously were available only by obtaining a different product edition.

This article describes the movement of a subset of PMEs to WebSphere Application Server for z/OS Version 6.1 from the following products:

- WebSphere Business Integration Server Foundation Version 5.1.x
- WebSphere Process Server for Multiplatforms Version 6

Overview of PME migration

The migration of PME services to WebSphere Application Server for z/OS Version 6.1 is handled on an individual basis. For PME services that are not supported in WebSphere Application Server for z/OS Version 6.1, all configuration information is removed. For PME services that are supported in WebSphere Application Server for z/OS Version 6.1, 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 Version 6.1. Any application that uses a resource for a service that is not supported by WebSphere Application Server Version 6.1 will not work correctly, and an error will be issued indicating that the application cannot be installed.

Running a mixed-node environment

When running in a mixed-node environment such as a WebSphere Application Server Version 6.1 deployment manager managing WebSphere Business Integration Server Foundation Version 5.1.x nodes, the first syncNode performed by the system downloads a configuration from the WebSphere Application Server Version 6.1 deployment manager to the WebSphere Business Integration Server Foundation Version 5.1.x nodes. Therefore, any PME service that is not supported by WebSphere Application Server Version 6.1 will be rendered inoperable on the WebSphere Business Integration Server Foundation Version 5.1.x node.

Chapter 5. Migrating product configurations

Use the migration tools to migrate WebSphere Application Server product configurations.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

The following configuration upgrades of WebSphere Application Server versions and offerings are directly supported.

Table 2. Directly Supported Configuration Upgrades

Migration Source	WebSphere Application Server for z/OS Version 6.1 Target		
	Stand-alone Application Server	Federated Application Server	Deployment Manager
WebSphere Application Server for z/OS Version 5.x or 6.0.x stand-alone application server	Supported		
WebSphere Application Server for z/OS Version 5.x or 6.0.x federated application server		Supported	
WebSphere Application Server for z/OS Version 5.x or 6.0.x deployment manager			Supported
WebSphere Business Integration Server Foundation for z/OS Version 5.1.x stand-alone application server	Supported *		
WebSphere Business Integration Server Foundation for z/OS Version 5.1.x federated application server		Supported *	
WebSphere Business Integration Server Foundation for z/OS Version 5.1.x deployment manager			Supported *

* WebSphere Application Server for z/OS Version 6.1 supports the migration of a subset of programming model extensions (PMEs) from WebSphere Business Integration Server Foundation. See “Programming model extension migration” on page 86 for more information on this support.

The migration utilities in WebSphere Application Server for z/OS Version 6.1 support migration from Version 5.x and Version 6.0.x. Use this article as a starting point for the planning information, Customization Dialog walk-throughs, and Version 5.x or 6.0.x to Version 6.1 migration explanations for stand-alone application server nodes, deployment managers, and federated nodes.

1. Migrate a stand-alone application server node to WebSphere Application Server Version 6.1.
See “Migrating stand-alone application servers” on page 91 for more information.
2. Migrate a deployment manager to WebSphere Application Server Version 6.1.
See “Migrating deployment managers” on page 106 for more information.
3. Migrate a federated node to WebSphere Application Server Version 6.1.
See “Migrating federated nodes” on page 116 for more information.

Configuration mapping during product-configuration migration

Various configurations are mapped during product-configuration migration.

Migration involves the copying of your configuration from a previous release of a WebSphere Application Server product into a new release.

Many migration scenarios are possible. The migration tools map objects and attributes existing in the version from which you are migrating to the corresponding objects and attributes in the Version 6.1 environment.

Bootstrap port

The migration tools map a non-default value directly into the Version 6.1 environment.

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 are not migrated because their roles in the WebSphere Application Server Version 6.1 configuration do not exist, have different meanings, or have different scopes.

Generic server

In WebSphere Application Server Version 5.1.x, a generic server was an APPLICATION_SERVER fitted to manage external resources. In Version 6.0.x and later, it has its own type called GENERIC_SERVER. Migration will perform this conversion, but migration cannot accurately migrate the external resources that the generic server references. After migration has completed migrating the generic server settings, you might need to perform additional tasks. If the old resource that the generic server was managing is located under the old WebSphere Application Server installation, perform the following tasks:

1. Copy any related files to the new installation.
2. Run any setup required to put the external application back into a valid and working state.

It is best that you reinstall the resource into the new WebSphere Application Server directory. Whatever you choose to do, the final step is to reset the reference to the new location of the application.

If the old resource that the generic server was managing is not installed under the old WebSphere Application Server installation, nothing further is required.

Migration of a Version 5.x or 6.0.x node to a Version 6.1 node

You can migrate a WebSphere Application Server Version 5.x or 6.0.x node that belongs to a cell without removing the node from the cell.

Migrate the deployment manager first, before migrating any base nodes in the cell.

Use the same cell name when migrating Network Deployment from Version 5.x or 6.0.x to Version 6.1. If you use a different cell name, federated nodes cannot successfully migrate to the Network Deployment Version 6.1 cell.

Migrating a base WebSphere Application Server node that is within a cell to Version 6.1 also migrates the node agent to Version 6.1. A cell can have some Version 6.1 nodes and other nodes that are at Version 5.x or 6.0.x levels. See “Coexistence support” on page 141 for information on restrictions on using mixed-release cells.

Policy file

WebSphere Application Server Version 6.1 migrates all the policy files that are installed with Version 5.x or 6.0.x by merging settings into the Version 6.1 policy files with the following characteristics:

- Any comments located in the Version 6.1 policy file will be preserved. Any comments contained in the Version 5.x or 6.0.x policy file will not be included in the Version 6.1 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 Version 6.1 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 and lib/app directories

Migration copies files from prior version directories into the WebSphere Application Server Version 6.1 configuration.

Property files

WebSphere Application Server Version 6.1 migrates all the property files that are installed with Version 5.x or 6.0.x by merging settings into the Version 6.1 property files with these exceptions for Version 5.x files:

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

Migration does not overlay property files.

Resource adapter archives (RARs) 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.

Migrating cluster-level resources:

WebSphere Application Server Version 6.0 introduced the concept of cluster-level resources. These are configured in resourcexxx.xml files under the cluster directories. For example:

```
<resources.j2c:J2CResourceAdapter xmi:id="J2CResourceAdapter_1112808424172"
  name="ims" archivePath="${WAS_INSTALL_ROOT}\installedConnectors\x2.rar">
  ...
</resources.j2c:J2CResourceAdapter>
```

If you have a cluster-level resource, this resource must be in the same location on each cluster member (node). Using the above example, therefore, each cluster member must have the RAR file installed at location `${WAS_INSTALL_ROOT}\installedConnectors\x2.rar`. `${WAS_INSTALL_ROOT}` is resolved on each cluster member to get the exact location.

In the migration of a deployment manager, the tools migrate the cluster files on the deployment manager, including the resourcexxx.xml files.

In the migration of a managed node, the tools process each J2C adapter. Files such as RAR files are migrated differently depending on whether you are migrating from Version 5.x to Version 6.x or from Version 6.0.x to Version 6.1.

- Version 5.x to Version 6.x migration
Because the profile structure changed in Version 6.0, migration from Version 5.x to Version 6.x copies files such as RAR files from the location set for the `WAS_INSTALL_ROOT` variable to the location set for the `USER_INSTALL_ROOT` variable and modifies any paths to reflect these changes.
- Version 6.0.x to Version 6.1 migration

Migration from Version 6.0.x to Version 6.1 copies files such as RAR files from WAS_INSTALL_ROOT to WAS_INSTALL_ROOT and from USER_INSTALL_ROOT to USER_INSTALL_ROOT.

If you have a RAR file in the WAS_INSTALL_ROOT for Version 6.0.x, for example, the migration tools do not automatically copy the file from WAS_INSTALL_ROOT to USER_INSTALL_ROOT as they would do in a Version 5.x to 6.x migration. This maintains the integrity of the cluster-level J2C resources.

Limitation: If you hardcoded a path to a RAR file (archivePath="C:/WAS/installedConnectors/x2.rar" for example) in Version 6.0.x, however, the Version 6.1 migration tools cannot change the archivePath attribute to reflect this because that would break all of the other cluster members that have not been migrated.

Samples

During the migration of the deployment manager, only WebSphere Application Server Version 5.x samples for federated nodes are migrated. Equivalent Version 6.1 samples are available for all other Version 5.x samples and all Version 6.0.x samples.

Security

Java 2 security is enabled by default when you enable security in WebSphere Application Server Version 6.1. Java 2 security requires you to grant security permissions explicitly.

When migrating to WebSphere Application Server Version 6.1, your choice of whether or not to migrate to support script compatibility results in one of two different outcomes.

- If you choose to migrate to support script compatibility, your security configuration is brought over to Version 6.1 without any changes.

This is the default.

- If you choose not to migrate to support script compatibility, the security configuration is converted to the default configuration for WebSphere Application Server Version 6.1. The default security configuration for Version 6.1 acts almost the same as in the previous versions, but there are some changes.

For example, existing keyfiles and trustfiles are moved out of the SSLConfig repertoire and new keystore and truststore objects are created.

All SSL configuration repertoires of the System Secure Sockets Layer (SSSL) type, except those that belong to the daemon, are converted to the Java Secure Socket Extension (JSSE) type.

For more information on migrating your security configurations to Version 6.1, see “Migrating, coexisting, and interoperating – Security considerations” on page 56.

Stdin, stdout, stderr, passivation, and working directories

In WebSphere Application Server for z/OS, outputs for stdin, stdout, and stderr are directed to SYSOUT by default. If they are redirected to the configuration directory of a previous version, you might need to change this in the Version 6.1 JCL.

The migration tools attempt to migrate existing passivation and working directories. Otherwise, appropriate Version 6.1 defaults are used.

If WebSphere Application Server for z/OS user IDs have home directories in the configuration directory of a previous version, you should update them before migration to reside in another location.

In a coexistence scenario, using common directories between versions can create problems.

Transport ports

The migration tools migrate all ports. The tools log a port-conflict warning if a port is already defined in the configuration. You must resolve any port conflicts before you can run servers at the same time.

Web modules

The specification level of the Java 2 Platform, Enterprise Edition (J2EE) implemented in WebSphere Application Server Version 6.0.x required 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 Web container no longer default to it but the Web container returns the call as "null." This situation might cause some browsers to display resulting Web container tags incorrectly. To prevent this problem from occurring, migration sets the `autoResponseEncoding` IBM extension to "true" for Web modules as it migrates enterprise applications.

Planning to migrate product configurations

You must prepare your system to migrate WebSphere Application Server for z/OS to Version 6.1.

WebSphere Application Server for z/OS supports the migration paths to Version 6.1 from the following releases:

- Version 5.0.x
- Version 5.1.x
- Version 6.0.x

In addition to migration, it is possible for the versions of the WebSphere Application Server mentioned above to coexist with Version 6.1, with the restriction that Version 5.0.x cannot coexist with Version 6.x in the same cell on the same LPAR. If you have multiple Version 5.0.x nodes on the same LPAR, all nodes must be migrated to Version 6.1 at the same time.

To learn more about the migration utilities, see "Overview of migration, coexistence, and interoperability" on page 77. When you are getting ready to migrate to Version 6.1, see "Premigration considerations" on page 81 and "Migrating stand-alone application servers," "Migrating Network Deployment configurations" on page 104, or "Migrating federated nodes" on page 116 for detailed planning and migration information.

1. Install the Version 6.1 product.
2. Review Chapter 4, "Migrating and coexisting," on page 77 to determine if you want the two versions of the product to coexist in the same environment or if you want to perform a complete migration.
3. Capture the variables used in the current installation.

This information will be useful in planning for the new installation, ensuring that you define unique ports, and planning for the configuration file system structure.

When you have finished planning and are ready to begin migrating, continue to Chapter 5, "Migrating product configurations," on page 87.

Migrating stand-alone application servers

The migration documentation for a stand-alone application server includes planning information, a Customization Dialog walk-through, a detailed Version 6.1 migration explanation, and a high-level migration checklist.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

Perform this task to migrate a WebSphere Application Server for z/OS Version 5.x or 6.0.x stand-alone application server to a Version 6.1 stand-alone application server.

Tip: For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

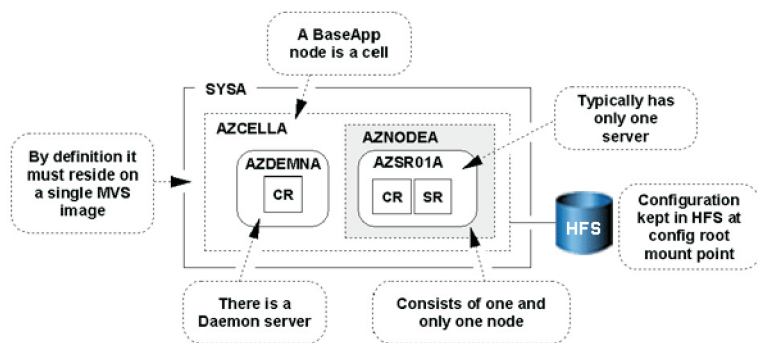
1. Print the “Checklist of activities for migrating a stand-alone application server” on page 100, and use it as you conduct your migration.
2. Inventory your applications and capture key information as described in “Preparing to migrate a stand-alone application server.”
3. Follow the instructions in “Customization Dialog walk-through for migrating a stand-alone application server” on page 94 to create a set of migration jobs.
4. Follow the instructions in “Migrating a stand-alone application server” on page 101 to migrate the stand-alone application server.

Preparing to migrate a stand-alone application server

Preparing to migrate a stand-alone application server node to WebSphere Application Server for z/OS Version 6.1 includes activities related to inventorying your existing configuration as well as generating CNTL and DATA data sets for the migration utilities.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

Migrating a stand-alone application server node is relatively simple because it consists of a single server and a daemon:



You might have configured multiple application servers inside a stand-alone application server node. Generally, this configuration is not recommended; but if you have this configuration, the migration utilities will migrate all servers in the node just as it would all the servers in a Network Deployment application server node.

WebSphere Application Server for z/OS Version 6.1 supplies five migration utilities in the data set that you specify during the **Allocate target data sets** step in the Customization Dialog:

- BBOWMG1B
- BBOWMG2B
- BBOMBHFS or BBOMBZFS
- BBOWMG3B
- BBOMBPCP

These are the jobs that you will use to perform the migration.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

1. Inventory your existing configuration.

- WebSphere Application Server for z/OS Version 5.x or 6.0.x configuration general information:

Cell short name:	
Cell long name:	
Sysplex name:	
Location of original Version 5.x or 6.0.x configuration data sets:	
WebSphere Application Server Version 5.x or 6.0.x home directory (configuration file system):	

- WebSphere Application Server for z/OS Version 5.x or 6.0.x stand-alone application server node:

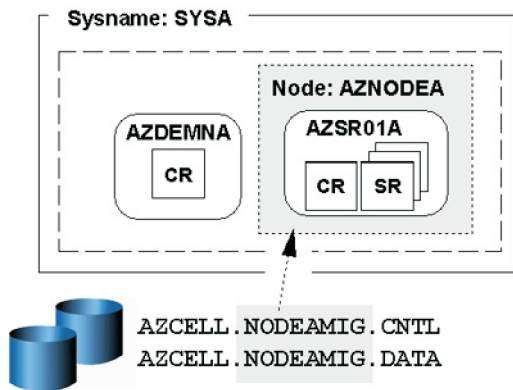
Configuration mount point:	
Configuration file system data set:	
Home directory:	Default: /AppServer Your value: _____
WebSphere Application Server SMP/E home:	Default: /usr/lpp/zWebSphere/V5R0M2 Your value: _____
Controller JCL start procedure:	
Servant JCL start procedure:	
Daemon JCL start procedure:	

- WebSphere Application Server for z/OS Version 6.1 installation information:

Product data set HLQ:	
Product directory (SMP/E target for Version 6.1):	
WebSphere Application Server Version 6.1 in LPA/LNKLST?:	<input type="checkbox"/> No <input type="checkbox"/> Yes

2. Produce migration CNTL/DATA for the node.

You must provide a set of CNTL and DATA data sets for the migration utilities to read in order to perform the migration. To do this, use the ISPF Customization Dialog for Version 6.1 to generate a set of customized jobs for the node in your configuration.



Some important points about these steps:

- You will be submitting these customized jobs
- When you go through the panels to generate these customized data sets, it will be necessary to supply information related to your configuration on most of the panels. This information is used to generate your customized jobs.

The purpose of creating these customized CNTL and DATA data sets is to give the migration utilities access to key information about your new Version 6.1 configuration (mount points and paths for example) as well as access to shell scripts customized with this new information. As illustrated here, the migration utilities get the bulk of the configuration information (settings and applications for example) from your existing Version 5.x or 6.0.x configuration file system.

For a detailed description of what you must do in the Customization Dialog to create your customized CNTL and DATA data sets, see “Customization Dialog walk-through for migrating a stand-alone application server.”

Customization Dialog walk-through for migrating a stand-alone application server

Walk through using the migration option within the Customization Dialog to generate the JCL jobs (CNTL and DATA data sets) for migrating a stand-alone application server node. You must create these jobs before you can begin the actual migration.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

Default values can be accepted for all fields and settings except for those that are explicitly identified in the following steps.

1. Invoke the WebSphere Application Server for z/OS Version 6.1 Customization Dialog.
ex 'product_hlq.SBBOCLIB(BBOWSTR)' 'options'
 - Example *product_hlq*: WAS610.WAS
 - Example *options*: appl(az)
2. From the first menu page, select option **4: Migrate a Node**.
3. On the next panel, select option **1: Migrate a stand-alone application server node**.
4. Allocate partitioned data sets that you will use to store the generated migration jobs and supporting data.

- a. On the **Stand-alone Application Server Migration** menu, select option 1: **Allocate target data sets**.
- b. On the **Allocate Target Data Sets** panel, specify your high-level qualifier and then press **Enter** to proceed to the next panel.
- c. Accept the defaults on the next two panels that specify the parameters for the .CNTL and .DATA data sets.

You are taken back to the **Stand-alone Application Server Migration** menu.

5. On the **Stand-alone Application Server Migration** menu, select option 2: **Define variables**.
In the following panels, the migration data collection process begins.
6. On the **Define Variables to Migrate a Stand-alone Application Server Node** menu, select option 1: **System locations (directories, HLQs, etc.)**.
7. On the **System Locations (1 of 2)** panel, specify your Version 6.1 installation libraries and whether you intend to put your Version 6.1 load modules in STEPLIB.

Correctly specifying STEPLIB is essential to a successful migration. It is likely that your Version 5.x or 6.0.x modules are currently in LPA/LNKLST and that you will begin with your Version 6.1 libraries being defined in STEPLIB.

Example:

System Locations (1 of 2)

Specify the following V6.1 information, then press ENTER to continue.

For some data sets, specify "Y" if they are in STEPLIB.

Full Names of Data Sets

PROCLIB.: SYS1.PROCLIB

Run WebSphere Application Server from STEPLIB (Y/N)? Y

SBBOLPA.: WAS610.WAS.SBBOLPA

SBBOLOAD: WAS610.WAS.SBBOLOAD

SBBGLOAD: WAS610.WAS.SBBGLOAD

SBBOLD2.: WAS610.WAS.SBBOLD2

Press **Enter** to proceed.

8. On the **System Locations (2 of 2)** panel, specify your Version 6.1 WebSphere Application Server product directory (/usr/lpp/zWebSphere/V6R1 for example).

Example:

System Locations (2 of 2)

Specify the following, then press Enter to continue.

V6.1 WebSphere Application Server product directory:
/usr/lpp/zWebSphere/V6R1

After specifying your Version 6.1 home directory, press **Enter** to proceed.

You are taken back to the **Define Variables to Migrate a Stand-alone Application Server Node** menu, where option 1 is marked as completed.

9. On the **Define Variables to Migrate a Stand-alone Application Server Node** menu, select option 2: **System environment customization**.
10. On the **System Environment Customization** panel, specify the Version 6.1 configuration file system.
The configuration file system is where the configuration for the node being migrated is physically stored. You can choose to use an existing Version 6.1 file system if you already have an appropriate file system on the node being migrated. If you choose to use an existing Version 6.1 file system, you need to ensure that the mount point that you specify here is present before running the migration utilities (BBOWMG1B, BBOWMG2B, and so on) that are created through these dialogs. If you choose

to create a new Version 6.1 file system on the node being migrated, the actual creation of the new file system will not occur until you run the BBOMBHFS or BBOMBZFS job during the migration process after you complete this walk-through. See “Migrating a stand-alone application server” on page 101 for more information.

Example:

System Environment Customization

Specify the following to customize your system environment, then press Enter to continue.

WebSphere Application Server for z/OS configuration file system information

Mount point....: /WebSphere/V6R1
Name.....: OMVS.WAS.CONFIG.HFS
Volume, or '*' for SMS.: *
Primary allocation in cylinders...: 250
Secondary allocation in cylinders.: 100

File system type (HFS or ZFS): HFS

After specifying either an existing mount point or a new mount point and the configuration file system type, press **Enter** to proceed.

You are taken back to the **Define Variables to Migrate a Stand-alone Application Server Node** menu, where options 1 and 2 are marked as completed.

11. On the **Define Variables to Migrate a Stand-alone Application Server Node** menu, select option **3: Server customization**.
12. On the **Server Customization (1 of 2)** panel, specify or leave the defaults for all values.
 - a. Specify the source node that you are migrating under **From WebSphere Application Server home directory**.
 - b. Specify the home location of the profile that will contain your Version 6.1 migrated node under **V6.1 WebSphere Application Server home directory**.
 - c. Specify the procedure names used to start the Version 6.1 servers.

Your Version 6.1 configuration must use different JCL procedures from those used by your Version 5.x or 6.0.x configuration. The migration process creates new Version 6.1 JCL procedures using the procedure names specified here.

If you want to use your existing procedure names, specify "N" in the **Replace Started Procedure Command Names** field.

Caution: If you use the same names as you used in your Version 5.x or 6.0.x configuration, the migration process overlays the existing procedures. If you are using the same names, make sure that you back up the existing Version 5.x or 6.0.x procedures before running the jobs generated by the Customization Dialog in case you need to roll back later.

Example:

Server Customization (1 of 2)

Specify the following to customize your migration, then press Enter to continue.

From WebSphere Application Server home directory:
/WebSphere/V5R1M0
/ AppServer

V6.1 WebSphere Application Server home directory:
/WebSphere/V6R1
/ AppServer

Replace Started Procedure Command Names: Y

Daemon Procedure name.....: BB06DMN

Controller Procedure name.....: BB06ACR

Servant Procedure name.....: BB06ASR

Adjunct Procedure name.....: BB06CRA

After you have specified or left the defaults for all values, press **Enter** to proceed to the next panel.

13. On the **Server Customization (2 of 2)** panel, specify the migration options and application migration settings.

- a. Specify whether or not to migrate to support script compatibility.

This field specifies whether (Y) or not (N) migration should create the following Version 5.x or 6.0.x configuration definitions:

- Transport
- ProcessDef
- Version 5.x or 6.0.x SSL
- Version 5.x or 6.0.x ORB service threadpool

instead of the following Version 6.1 configuration definitions:

- Channels
- ProcessDefs
- Version 6.1 SSL
- Version 6.1 ORB service threadpool

Choose to migrate to support script compatibility if you want to minimize impacts to existing administration scripts. If you have existing **wsadmin** scripts or programs that use third-party configuration APIs to create or modify the Version 5.x or 6.0.x configuration definitions, for example, you might want to choose this option during migration.

Note: This is meant to provide a temporary transition until all of the nodes in the environment are at the Version 6.1 level. When they are all at the Version 6.1 level, you should perform the following actions:

- 1) Modify your administration scripts to use all of the Version 6.1 settings.
- 2) Use the **convertScriptCompatibility** command to convert your configurations to match all of the Version 6.1 settings.

See “convertScriptCompatibility command” on page 129.

- b. Specify whether or not to enable tracing on the migration utilities.

If you choose to enable tracing, it will remain enabled throughout the entire migration process.

- c. Specify the location of the temporary directory for the migration configuration backup, temporary, output, and error files.

During migration, a backup copy of the Version 5.x or 6.0.x configuration is required. The default location of this backup is already specified, though you can override if needed. You might need to specify a location other than the default if the /tmp file system does not have adequate space to store the backup configuration. If you choose to override the default location of the backup copy, the best practice is to keep the same naming convention and just replace the **/tmp** portion with another path, **/myTemp/migrate** for example.

- d. Note the migration identifier.

The five-digit migration identifier (55449 in this example) is generated uniquely each time you create the migration jobs. The migration output messages, which you will need to monitor throughout the migration process, are stored in /tmp/migrate/55449. (After migration, the job

output in this directory is NOT automatically deleted.) The migration output messages are also appended to the JCL sysout messages, which can be viewed in SDSF.

- e. Specify your preference for the migration of applications.

Choose one of the following options:

- Y** Install user enterprise applications in the Application Installation Directory as part of the migration.
- S** Prepare user enterprise applications for installation in the WebSphere Application Server Version 6.1 installableApps directory without actually installing them during migration.
- JACL scripts that can be used to install these applications are generated and saved in the migration backup directory. You can then run these files at any point and in any combination after migration has completed. You can also reorganize and combine these JACL files for better applications installation efficiency if you want.
- P** Install user enterprise applications as part of the migration, and keep the same application installation directories as the previous version.

Restrictions: If you select this option, the location is shared by the existing WebSphere Application Server Version 5.x or 6.0.x installation and the Version 6.1 installation. If you keep the migrated applications in the same locations as those of the previous version, the following restrictions apply:

- The WebSphere Application Server Version 6.1 mixed-node support limitations must be followed. This means that the following support cannot be used when evoking the **wsadmin** command:
 - Precompile JSP
 - Use Binary Configuration
 - Deploy EJB
- You risk losing the migrated applications unintentionally if you later delete applications from these locations when administering (uninstalling for example) your Version 5.x or 6.0.x installation.

- N** Do nothing with user enterprise applications.

WebSphere Application Server system applications will migrate regardless of the value set here.

- f. Specify the directory in which to install the applications if you choose to install user enterprise applications in the Application Installation Directory as part of the migration.

Example:

Server Customization (2 of 2)

Specify the following to customize your migration, then press Enter to continue.

Migration Options

Migrate to support script compatibility: Y
Enable Tracing: Script: N Profile: N PreUpgrade: N PostUpgrade: N
Temporary Directory Location: /tmp/migrate
Migration Identifier.....: 55449

Application Migration Settings:

Application Migration Preference: Y (Y/S/P/N)
(Y) Migrate and install applications using the Application Installation Directory.
(S) Migrate Applications and generate scripts for later installation.
(P) Migrate Applications and keep the same application installation

directories as the previous version.
(N) Do not migrate applications.

Application Installation Directory:
==> /WebSphere/V6R1/AppServer/profiles/default/installedApps

Specify values or accept defaults, and press **Enter** to proceed.

You are returned to the **Define Variables to Migrate a Stand-alone Application Server Node** menu.

14. Press **PF3** to return to the **Stand-alone Application Server Migration** menu.
15. On the **Stand-alone Application Server Migration** menu, select option **3: Generate customization jobs**.
16. On the **Generate Customization Jobs** panel, provide a job card customized to your environment-specific requirements.

On the **Generate Customization Jobs** panel, you can see the names of the .CNTL and .DATA data sets that have been customized based on your previous input.

Example:

Generate Customization Jobs

This portion of the Customization Dialog generates the jobs you must run after you complete this dialog process. You must complete the customization process before you generate the jobs with this step. If you have not done this, please return to that step.

Jobs and data files will get generated into data sets:

```
'hlq.CNTL'  
'hlq.DATA'
```

If you wish to generate customization jobs using other data sets, then exit from this panel and select the "Allocate target data sets" option.

All the jobs that will be tailored for you will need a job card. Please enter a valid job card for your installation below. The file tailoring process will update the job name for you in all the generated jobs, so you need not be concerned with that portion of the job cards below. If continuations are needed, replace the comment cards with continuations.

Specify the job cards, then press Enter to continue.

```
//jobname JOB (ACCTNO,ROOM),'IBMUSER',CLASS=A,REGION=0M  
//*  
//*  
//*
```

After you have entered an appropriate job card, press **Enter**.

The generation process runs and presents you with a list of job streams and files that have been created to perform the migration to Version 6.1. These jobs do not require any editing; they are to be submitted "as-is" during the migration process.

17. When processing has completed, press **Enter** to proceed.

Example:

```
Processing for data set 'hlq.CNTL' ...  
Member BBOWMG1B successfully created.  
Member BBOWMG2B successfully created.  
Member BBOWMG3B successfully created.  
Member BBOMBHFS successfully created.  
Member BBOMBCCR successfully created.  
Member BBOMBCCRZ successfully created.  
Member BBOMBBDN successfully created.
```

Member BBOMBDNZ successfully created.
 Member BBOMBSR successfully created.
 Member BBOMBSRZ successfully created.
 Member BBOMBBCP successfully created.
 Member BBOMBINS successfully created.

Processing for data set 'hlq.DATA' ...
 Member BBOWBMPT successfully created.
 Member BBOWMBRF successfully created.

18. On the **Stand-alone Application Server Migration** menu, save your configuration variables for future reference by selecting option **S: Save customization variables**.

You have now completed the job generation process and are ready to begin the process of migration.

19. See the instructions for performing the migration by selecting option **4: View instructions**.

Also, see “Migrating a stand-alone application server” on page 101 for additional instructions on performing the migration.

Checklist of activities for migrating a stand-alone application server

Activities for migrating a stand-alone application server after you have generated the appropriate JCL jobs using the Customization Dialog include activities related to general preparation, transaction-log clearance, and node migration.

This checklist is intended to provide a quick reference.

More details can be found under “Preparing to migrate a stand-alone application server” on page 92.

Key similarities to migrating a deployment manager:

- You must still create CNTL/DATA data sets using the Customization Dialog.
- You must still allocate and mount a new configuration file system to act as the target for the new Version 6.1 configuration, either manually or using BBOMBHFS or BBOMBZFS.
- You must still run BBOWMG3B to perform the migration.

Key differences from migrating a deployment manager:

- You must clear the transaction logs using BBOWMG1B; you do not have to do this for a deployment manager migration.
- You must disable PRR mode using BBOWMG2B; you do not have to do this for a deployment manager migration.
- There is no deployment manager node to migrate.

Prepare to migrate the application server

Check off	Item
	Make sure that you have a minimum level of Version 6.1 maintenance. See “Overview of migration, coexistence, and interoperability” on page 77.
	Inventory your existing Version 5.x or 6.0.x configuration and capture key information. See “Preparing to migrate a stand-alone application server” on page 92.
	Create CNTL/DATA data sets for each node to be migrated. See “Customization Dialog walk-through for migrating a stand-alone application server” on page 94).
	Create Version 6.1 configuration mount point; allocate configuration file system, and mount either manually or using BBOMBHFS or BBOMBZFS.
	Run BBOMBBCP to modify the JCL start procedure names.

Clear the transaction logs for the application server node

Check off	Item
	Stop the server in the stand-alone application server node.
	Run BBOWMG1B, and check for success.
	Restart the application server to clear the transaction logs. It will start and then automatically stop.
	Run BBOWMG2B to disable PRR mode.

Migrate the application server

Check off	Item
	Ensure that all servers are stopped, including the daemon server.
	Run BBOWMG3B, and check for success.
	Start the application server.

Migrating a stand-alone application server

After you use the Customization Dialog to generate JCL jobs for migrating a stand-alone application server node to WebSphere Application Server for z/OS Version 6.1, you can perform the actual migration by running those jobs. For help in running these jobs, see the instructions that were generated when you generated the jobs.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

Make sure that you go through the steps outlined in “Customization Dialog walk-through for migrating a stand-alone application server” on page 94 before you begin this migration. You will not be able to proceed if you have not created the JCL migration utilities through the Customization Dialog.

The BBOWMG1B, BBOWMG2B, and BBOWMG3B jobs referenced below must be submitted by a WebSphere Administrator User ID. All other jobs must be submitted by a user ID that has control over the file system.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

Tip: Before migrating a WebSphere Application Server Version 5.x or 6.0.x stand-alone application server, use the **backupConfig** command or your own preferred backup utility to back up your existing configuration if you want to be able to restore it to its previous state after migration. Make sure that you note the exact name and location of this backed-up configuration.

1. Create and mount a new Version 6.1 configuration file system.

Before you perform the migration, Version 6.1 requires a file system to be present for your new configuration. You can run BBOMBHFS or BBOMBZFS to create and mount a new configuration file system, or you can mount one manually. Either way, you must have a file system for your Version 6.1 configuration created and mounted before you proceed. This configuration file system is the target of the migration; your Version 5.x or 6.0.x configuration file system is the source.

BBOMBHFS or BBOMBZFS creates a mount point directory, allocates the configuration’s file system, and mounts the file system at whatever value you specified for **Mount point** on the **System Environment Customization** panel in the Customization Dialog.

Ensure that you have allocated, created, and mounted your configuration file system data sets either manually or using BBOMBHFS or BBOMBZFS before you proceed. The mount point should be owned

by the WebSphere Admin ID and have permissions of at least 755. The new configuration file system structures should be included in BPXPARM so that they will be mounted at the next IPL.

2. Copy your generated JCL procedures.

The migration utility BBOMBCP copies the generated JCL procedures used to start the servers to the specified procedure library. Your Version 6.1 configuration must use different JCL procedures from those used by your Version 5.x or 6.0.x configuration. This utility will update the new Version 6.1 configuration, substituting your new JCL names in place of the names that existed in your original Version 5.x or 6.0.x configuration.

Caution: This utility copies the dialog-generated JCL to your procedure library. If you specified the same names in the Customization Dialog as you used in your Version 5.x or 6.0.x configuration, this utility will overlay the existing procedures. If you are using the same names, make sure that you back up the existing Version 5.x or 6.0.x procedures before running this utility in case you need to roll back later.

Submit BBOMBCP, and verify a return code of 0.

3. If you specified new procedure names, update your RACF STARTED profiles for the controller and daemon.

The STARTED profile used by controller regions is based on the procedure name and JOBNAME. You must ensure that a STARTED profile will apply so that the proper identity will be assigned to the started task. If your Version 5.x or 6.0.x controller JCL procedure name is AZACR and you specified AZ6ACR for Version 6.1, for example, then you would need to create a STARTED profile for that new procedure name:

new controller	same identity used in
JCL name	V5.x or V6.0.x configuration
RDEFINE STARTED AZ6ACR.*	STDATA(USER(AZACRU) GROUP(AZCFG) TRACE(YES))

Note:

- Do not use a different user ID to start. There are other things tied to the user ID; and if you change the user ID, other changes would also be required.
- If your original STARTED profile was generic, STARTED AZ*.* ... for example, you would not need to create a new STARTED profile.
- Servant region STARTED profiles are based on JOBNAME, not procedure name. So there is no issue with the servant when you use a different procedure name.
- Daemons and node agents are controllers, so using different procedure names for those implies a new STARTED profile.

4. Submit BBOWMG1B and BBOWMG2B.

Note: If you are not using XA connectors, submitting BBOWMG1B and BBOWMG2B is optional. However, you should submit both jobs to ensure that your transaction logs are clear.

BBOWMG1B enables all servers on the application server node being migrated to start in PRR processing mode. PRR processing mode resolves any outstanding transactions, clears the transaction logs, and terminates the server. BBOWMG2B disables PRR mode and returns all servers to normal operating state.

Follow these steps to clear the XA transaction logs:

- a. Submit the job BBOWMG1B, and verify a return code of 0.
 - b. Restart the application server, and wait for it to perform PRR processing and terminate automatically.
 - c. Submit the job BBOWMG2B, and verify a return code of 0.
5. Stop the Version 5.x or 6.0.x daemon and application server.

The daemon must be at the highest version level of any of the servers that it manages on the same LPAR. It will be at the Version 6.1 level when the server is started.

You must stop the Version 5.x or 6.0.x application server before proceeding.

6. Delete and redefine the log stream.

Perform this step only if you previously configured the transaction XA partner log or compensation log on the Version 5.x or 6.0.x server to use a log stream.

- a. Make sure that the node is shut down.
- b. Delete and redefine the log stream.

You can use the BBOLOGSD and BBOLOGSA jobs that were created during Version 5.x or 6.0.x customization if you configured the server initially to use the log stream.

The following sample shows an example of such a job:

```
//RLSP1A JOB 'xxxx,yyy,?', 'USERID',MSGCLASS=H,
//          CLASS=J,MSGLEVEL=(1,1),REGION=4M,NOTIFY=&SYSUID
//STEP1 EXEC PGM=IXCMIAPU
//STEPLIB DD DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *

DATA TYPE(LOGR) REPORT(YES) /* Default to show output of job */
DELETE LOGSTREAM NAME(P1ACEL6A.W51ASA2.D)
DEFINE LOGSTREAM NAME(P1ACEL6A.W51ASA2.D)
        LOWOFFLOAD(20)
        HIGHOFFLOAD(79)
        STG_DUPLEX(YES)
        DUPLEXMODE(UNCOND)
        STG_DATACLAS(OPERLOG)
        STG_SIZE(5000)
        HLQ(Q10RRS)
        LS_SIZE(5000)
        LS_DATACLAS(OPERLOG)
        STRUCTNAME(WAS_LOGRLS)

/*
```

7. Submit BBOWMG3B.

BBOWMG3B is the job that performs the physical migration of the Version 5.x or 6.0.x node to Version 6.1 based on the information that you supplied in the Customization Dialog. Submit BBOWMG3B, verify that you are getting return codes of 0, and review the log files in the migration temporary directory on the file system. The migration temporary directory is *temporary_directory_location/nnnnn*, where *temporary_directory_location* is the directory specified for **Temporary Directory Location** (/tmp/migrate by default) and *nnnnn* is the numeric value specified for **Migration identifier** on the **Server Customization (2 of 2)** panel in the Customization Dialog.

8. Start the new application server node.

Use the existing commands that you would use to start a Version 5.x or 6.0.x application server, but replace the RACF STARTED procedure name with the value that you entered in the **Server Customization (1 of 2)** panel for **Controller procedure name** in the Customization Dialog. This command starts the Version 6.1 application server. Wait until the server is finished initializing before proceeding.

The following message is displayed on the console and in the job log of BBOS001:

```
BB000019I INITIALIZATION COMPLETE FOR WEBSPHERE FOR z/OS CONTROL PROCESS BBOS001
```

At this point, your migration to Version 6.1 is complete.

9. Perform the post-migration tasks.

After you have verified a successful migration to Version 6.1 and are successfully running a migrated configuration, you should delete the following items:

- Everything in the source configuration's file system
- Everything in the target configuration's *temporary_directory_location/nnnnn* directory, where *temporary_directory_location* is the directory specified for **Temporary Directory Location**

(/tmp/migrate by default) and *nnnn* is the numeric value specified for **Migration identifier** on the **Server Customization (2 of 2)** panel in the Customization Dialog

- The `bbomigt2.sh` file

Migrating Network Deployment configurations

The migration documentation for Network Deployment configurations includes planning information, a Customization Dialog walk-through, a detailed Version 6.1 migration explanation, and a high-level migration checklist.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

Select the appropriate link for information on how to migrate to a WebSphere Application Server for z/OS Version 6.1 deployment manager.

- “Preparing to migrate a Network Deployment configuration”
- “Migrating deployment managers” on page 106
- “Migrating federated nodes” on page 116

Tip: For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

Preparing to migrate a Network Deployment configuration

Preparing to migrate a Network Deployment configuration to WebSphere Application Server for z/OS Version 6.1 includes activities related to inventorying your existing configuration, planning your node-by-node strategy, and generating CNTL and DATA data sets for the migration utilities.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

WebSphere Application Server for z/OS Version 6.1 supplies migration utilities in the data set that you specify during the **Allocate target data sets** step in the Customization Dialog:

- BBOMDHFS or BBOMDZFS
- BBOWMG3D
- BBOMDCP

These are the jobs that you will use to perform the migration.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

1. Inventory your existing configuration.

- WebSphere Application Server for z/OS Version 5.x or 6.0.x configuration general information:

Cell short name:	
Cell long name:	
Sysplex name:	
Configuration file system strategy:	<input type="checkbox"/> All nodes share one file system <input type="checkbox"/> Unique file system per node
JCL start procedure strategy:	<input type="checkbox"/> Use common JCL <input type="checkbox"/> Unique JCL per server

Location of original Version 5.x or 6.0.x configuration data sets:	
WebSphere Application Server Version 5.x or 6.0.x home directory (configuration file system):	

Note: You also need a WebSphere Application Server for z/OS administrator user ID and password.

- WebSphere Application Server for z/OS Version 5.x or 6.0.x Deployment Manager node:

Configuration mount point:	
Configuration file system data set:	
Home directory:	Default:/DeploymentManager Your value: _____
WebSphere Application Server SMP/E home:	Default: /usr/lpp/zWebSphere/V6R1 Your value: _____
Controller JCL start procedure:	
Servant JCL start procedure:	
Daemon JCL start procedure:	

- WebSphere Application Server for z/OS Version 6.1 installation information:

Product data set HLQ:	
Product directory (SMP/E target for Version 6.1):	
WebSphere Application Server Version 6.1 in LPA/LNKLST?:	<input type="checkbox"/> No <input type="checkbox"/> Yes

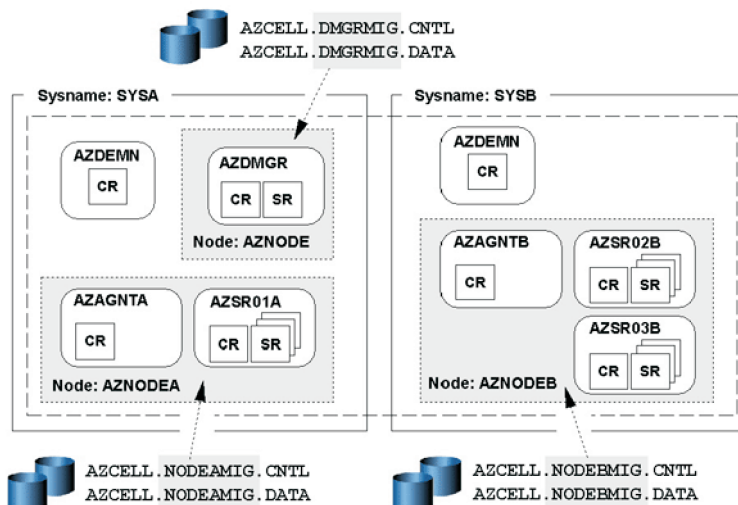
2. Plan the node-by-node migration strategy.

Take a moment to map out your migration sequence:

	Node Short Name	System Name
First node		
Second node		
Third node		
Fourth node		
Fifth node		
Sixth node		

3. Produce migration CNTL/DATA for each node.

You must provide a set of CNTL and DATA data sets for the migration utilities to read in order to perform the migration. To do this, use the ISPF Customization Dialog for Version 6.1 to generate a set of customized jobs for each node in your configuration.



Some important points about these steps:

- You will be submitting these customized jobs
- When you go through the panels to generate these customized data sets, it will be necessary to supply information related to your configuration on most of the panels. This information is used to generate your customized jobs.

The purpose of creating these customized CNTL and DATA data sets is to give the migration utilities access to key information about your new Version 6.1 configuration (mount points and paths for example) as well as access to shell scripts customized with this new information. As illustrated here, the migration utilities get the bulk of the configuration information (settings and applications for example) from your existing Version 5.x or 6.0.x configuration file system.

For a detailed description of what you must do in the Customization Dialog to create your customized CNTL and DATA data sets, see “Customization Dialog walk-through for migrating a deployment manager” on page 107.

Migrating deployment managers

The migration documentation for deployment managers includes a detailed WebSphere Application Server Version 6.1 migration explanation, a Customization Dialog walk-through, and a high-level migration checklist.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

Perform this task to migrate a WebSphere Application Server for z/OS Version 5.x or 6.0.x deployment manager to a Version 6.1 deployment manager.

Tip: For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

1. Print the “Checklist of activities for migrating a deployment manager” on page 107, and use it as you conduct your migration.
2. Inventory your applications and capture key information as described in “Preparing to migrate a Network Deployment configuration” on page 104.
3. Follow the instructions in “Customization Dialog walk-through for migrating a deployment manager” on page 107 to create a set of migration jobs.
4. Follow the instructions in “Migrating a deployment manager” on page 113 to migrate the deployment manager.

Checklist of activities for migrating a deployment manager

Activities for migrating a deployment manager after you have generated the appropriate JCL jobs using the Customization Dialog include activities related to general preparation and node migration.

This checklist is intended to provide a quick reference.

More details can be found here: “Preparing to migrate a Network Deployment configuration” on page 104.

Prepare to migrate the deployment manager

Check off	Item
	Inventory your existing Version 5.x or 6.0.x configuration and capture key information.
	Map a strategy for the sequence of nodes to be migrated.
	Create CNTL/DATA data sets for each node to be migrated using the Customization Dialog. See “Customization Dialog walk-through for migrating a deployment manager.”
	Create the Version 6.1 configuration mount point, allocate configuration file system, and mount either manually or using BBOMDHFS or BBOMDZFS.
	Customize and run BBOMDCP to modify the JCL start procedure names used.

Migrate the deployment manager

Check off	Item
	Stop the deployment manager and the associated daemon server.
	Run BBOWMG3D, and check the output for success.
	Customize and run BBOWBPR to modify the JCL start procedure names used.
	Start the Version 6.1 copy of the deployment manager.

Customization Dialog walk-through for migrating a deployment manager

Walk through using the Customization Dialog to generate the JCL jobs (CNTL and DATA data sets) for migrating your deployment manager. You must create these jobs before you can begin the actual migration.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

Default values can be accepted for all fields and settings except for those that are explicitly identified in the following steps.

1. Invoke the WebSphere Application Server for z/OS Version 6.1 Customization Dialog.

ex `'product_hlq.SBBOCLIB(BBOWSTRT)' 'options'`

- Example `product_hlq`: WAS610.WAS
- Example `options`: appl(az)

For more information on starting the Customization Dialog, including information on available options, see the *Installing your application serving environment* PDF.

2. From the first menu page, select option **4: Migrate a Node**.
3. On the next panel, select option **2: Migrate a deployment manager**.
4. Allocate partitioned data sets that you will use to store the generated migration jobs and supporting data.

- a. On the **Deployment Manager Migration** menu, select option 1: **Allocate target data sets**.
- b. On the **Allocate Target Data Sets** panel, specify your high-level qualifier and then press **Enter** to proceed to the next panel.
- c. Accept the defaults on the next two panels that specify the parameters for the .CNTL and .DATA data sets.

You are taken back to the **Deployment Manager Migration** menu.

5. On the **Deployment Manager Migration** menu, select option 2: **Define variables**.
In the following panels, the migration data collection process begins.
6. On the **Define Variables to migrate a Deployment Manager Node** menu, select option 1: **System locations (directories, HLQs, etc.)**.
7. On the **System Locations (1 of 2)** panel, specify your Version 6.1 installation libraries and whether you intend to put your Version 6.1 load modules in STEPLIB.

Correctly specifying STEPLIB is essential to a successful migration. It is likely that your Version 5.x or 6.0.x modules are currently in LPA/LNKLST and that you will begin with your Version 6.1 libraries being defined in STEPLIB.

For more information on STEPLIB, see the *Installing your application serving environment* PDF.

Example:

System Locations (1 of 2)

Specify the following V6.1 information, then press ENTER to continue.

For some data sets, specify "Y" if they are in STEPLIB.

Full Names of Data Sets

PROCLIB.: SYS1.PROCLIB

Run WebSphere Application Server from STEPLIB (Y/N)? Y

SBBOLPA.: WAS610.WAS.SBBOLPA

SBBLOAD.: WAS610.WAS.SBBLOAD

SBBGLOAD.: WAS610.WAS.SBBGLOAD

SBBOLD2.: WAS610.WAS.SBBOLD2

Press **Enter** to proceed.

8. On the **System Locations (2 of 2)** panel, specify your Version 6.1 WebSphere Application Server product directory (/usr/lpp/zWebSphere/V6R1 for example).

Example:

System Locations (2 of 2)

Specify the following, then press Enter to continue.

V6.1 WebSphere Application Server product directory:

/usr/lpp/zWebSphere/V6R1

After specifying your Version 6.1 home directory, press **Enter** to proceed.

You are taken back to the **Define Variables to migrate a Deployment Manager Node** menu, where option 1 is marked as completed.

9. On the **Define Variables to migrate a Deployment Manager Node** menu, select option 2: **System environment customization**.
10. On the **System Environment Customization** panel, specify the Version 6.1 configuration file system.
The configuration file system is where the configuration for the node being migrated is physically stored. You can choose to use an existing Version 6.1 file system if you already have an appropriate file system on the node being migrated. If you choose to use an existing Versions 6.1 file system, you need to ensure that the mount point that you specify here is present before running the migration

utilities (BBOMDHFS or BBOMDZFS, BBOMDCP, and so on) that are created through these dialogs. If you choose to create a new Version 6.1 file system on the node being migrated, the actual creation of the new file system will not occur until you run the BBOMDHFS or BBOMDZFS job during the migration process after you complete this walk-through. See “Migrating a deployment manager” on page 113 for more information.

Example:

System Environment Customization

Specify the following to customize your system environment, then press Enter to continue.

WebSphere Application Server for z/OS configuration file system information

```
Mount point....: /WebSphere/V6R1
Name.....: OMVS.WAS.CONFIG.HFS
Volume, or '*' for SMS.: *
Primary allocation in cylinders...: 250
Secondary allocation in cylinders.: 100
```

File system type (HFS or ZFS): HFS

After specifying either an existing mount point or a new mount point and the configuration file system type, press **Enter** to proceed.

You are taken back to the **Define Variables to migrate a Deployment Manager Node** menu, where options 1 and 2 are marked as completed.

11. On the **Define Variables to migrate a Deployment Manager Node** menu, select option **3: Server customization**.

12. On the **Server Customization (1 of 2)** panel, specify or leave the defaults for all values.

- a. Specify the source node that you are migrating under **From WebSphere Application Server home directory**.
- b. Specify the home location of the profile that will contain your Version 6.1 migrated node under **V6.1 WebSphere Application Server home directory**.
- c. Specify the procedure names used to start the Version 6.1 servers.

Your Version 6.1 configuration must use different JCL procedures from those used by your Version 5.x or 6.0.x configuration. The migration process creates new Version 6.1 JCL procedures using the procedure names specified here.

If you want to use your existing procedure names, specify "N" in the **Replace Started Procedure Command Names** field.

Caution: If you use the same names as you used in your Version 5.x or 6.0.x configuration, the migration process overlays the existing procedures. If you are using the same names, make sure that you back up the existing Version 5.x or 6.0.x procedures before running the jobs generated by the Customization Dialog in case you need to roll back later.

Example:

Server Customization (1 of 2)

Specify the following to customize your migration, then press Enter to continue.

```
From WebSphere Application Server home directory:
  /WebSphere/V5R1M0
  / DeploymentManager
```

```
V6.1 WebSphere Application Server home directory:
  /WebSphere/V6R1
```

/ DeploymentManager

Replace Started Procedure Command Names: Y

Daemon Procedure name.....: BB06DMN

Controller Procedure name....: BB06DCR

Servant Procedure name.....: BB06DSR

After you have specified or left the defaults for all values, press **Enter** to proceed to the next panel.

13. On the **Server Customization (2 of 2)** panel, specify the migration options and application migration settings.

- a. Specify whether or not to migrate to support script compatibility.

This field specifies whether (Y) or not (N) migration should create the following Version 5.x or 6.0.x configuration definitions:

- Transport
- ProcessDef
- Version 5.x or 6.0.x SSL
- Version 5.x or 6.0.x ORB service threadpool

instead of the following Version 6.1 configuration definitions:

- Channels
- ProcessDefs
- Version 6.1 SSL
- Version 6.1 ORB service threadpool

Choose to migrate to support script compatibility if you want to minimize impacts to existing administration scripts. If you have existing **wsadmin** scripts or programs that use third-party configuration APIs to create or modify the Version 5.x or 6.0.x configuration definitions, for example, you might want to choose this option during migration.

Note: This is meant to provide a temporary transition until all of the nodes in the environment are at the Version 6.1 level. When they are all at the Version 6.1 level, you should perform the following actions:

- 1) Modify your administration scripts to use all of the Version 6.1 settings.
- 2) Use the **convertScriptCompatibility** command to convert your configurations to match all of the Version 6.1 settings.

See “convertScriptCompatibility command” on page 129.

- b. Specify whether or not to disable the previous deployment manager during migration.

If the Version 5.x or 6.0.x deployment manager is not disabled, you can continue to use it while the migration is completed.

Caution: Use this option with care.

- The reason that WebSphere Application Server Version 5.x or 6.0.x deployment manager configurations normally are stopped and disabled is to prevent multiple deployment managers from managing the same nodes. You must stop the Version 5.x or 6.0.x deployment manager before you start using the Version 6.1 deployment manager. The most likely error conditions that will occur if this is not done are port conflicts when the second instance of the deployment manager is started.
- Specifying no (N) for this field means that any configuration changes made in the old configuration during migration might not be migrated.

- c. Specify whether or not to enable tracing on the migration utilities.

If you choose to enable tracing, it will remain enabled throughout the entire migration process.

- d. Specify the location of the temporary directory for the migration configuration backup, temporary, output, and error files.

During migration, a backup copy of the Version 5.x or 6.0.x configuration is required. The default location of this backup is already specified, though you can override if needed. You might need to specify a location other than the default if the /tmp file system does not have adequate space to store the backup configuration. If you choose to override the default location of the backup copy, the best practice is to keep the same naming convention and just replace the /tmp portion with another path, /myTemp/migrate for example.

- e. Note the migration identifier.

The five-digit migration identifier (53446 in this example) is generated uniquely each time you create the migration jobs. The migration output messages, which you will need to monitor throughout the migration process, are stored in /tmp/migrate/53446. (After migration, the job output in this directory is NOT automatically deleted.) The migration output messages are also appended to the JCL sysout messages, which can be viewed in SDSF.

- f. Specify your preference for the migration of applications.

Choose one of the following options:

Y Install user enterprise applications in the Application Installation Directory as part of the migration.

S Prepare user enterprise applications for installation in the WebSphere Application Server Version 6.1 installableApps directory without actually installing them during migration.

JACL scripts that can be used to install these applications are generated and saved in the migration backup directory. You can then run these files at any point and in any combination after migration has completed. You can also reorganize and combine these JACL files for better applications installation efficiency if you want.

P Install user enterprise applications as part of the migration, and keep the same application installation directories as the previous version.

Restrictions: If you select this option, the location is shared by the existing WebSphere Application Server Version 5.x or 6.0.x installation and the Version 6.1 installation. If you keep the migrated applications in the same locations as those of the previous version, the following restrictions apply:

- The WebSphere Application Server Version 6.1 mixed-node support limitations must be followed. This means that the following support cannot be used when evoking the **wsadmin** command:
 - Precompile JSP
 - Use Binary Configuration
 - Deploy EJB
- You risk losing the migrated applications unintentionally if you later delete applications from these locations when administering (uninstalling for example) your Version 5.x or 6.0.x installation.

N Do nothing with user enterprise applications.

WebSphere Application Server system applications will migrate regardless of the value set here.

- g. Specify the directory in which to install the applications if you choose to install user enterprise applications in the Application Installation Directory as part of the migration .

- h. Specify an administrator user ID and password.

Because the migration jobs perform administrative functions, you need to supply a valid administrator user ID and password here.

Example:

Server Customization (2 of 2)

Specify the following to customize your migration, then press Enter to continue.

Migration Options

Migrate to support script compatibility: Y
Disable Previous Deployment Manager: Y
Enable Tracing: Script: N Profile: N PreUpgrade: N PostUpgrade: N
Temporary Directory Location: /tmp/migrate
Migration Identifier.....: 53446

Application Migration Settings:

Application Migration Preference: Y (Y/S/P/N)
(Y) Migrate and install applications using the Application Installation Directory.
(S) Migrate Applications and generate scripts for later installation.
(P) Migrate Applications and keep the same application installation directories as the previous version.
(N) Do not migrate applications.

Application Installation Directory:
==> /WebSphere/V6R1/DeploymentManager/profiles/default/installedApps

WebSphere Administrator User ID.: XXXXXX
WebSphere Administrator Password: XXXXXXXX

Specify values or accept defaults, and press **Enter** to proceed.

You are returned to the **Define Variables to migrate a Deployment Manager Node** menu.

14. Press **PF3** to return to the **Deployment Manager Migration** menu.
15. On the **Deployment Manager Migration** menu, select option **3: Generate customization jobs**.
16. On the **Generate Customization Jobs** panel, provide a job card customized to your environment-specific requirements.

On the **Generate Customization Jobs** panel, you can see the names of the .CNTL and .DATA data sets that have been customized based on your previous input.

Example:

Generate Customization Jobs

This portion of the Customization Dialog generates the jobs you must run after you complete this dialog process. You must complete the customization process before you generate the jobs with this step. If you have not done this, please return to that step.

Jobs and data files will get generated into data sets:
'hlq.CNTL'
'hlq.DATA'

If you wish to generate customization jobs using other data sets, then exit from this panel and select the "Allocate target data sets" option.

All the jobs that will be tailored for you will need a job card. Please enter a valid job card for your installation below. The file tailoring process will update the job name for you in all the generated jobs, so you need not be concerned with that portion of the job cards below. If continuations are needed, replace the comment cards with continuations.

Specify the job cards, then press Enter to continue.

```
//jobname JOB (ACCTNO,ROOM),'IBMUSER',CLASS=A,REGION=0M
//*
//*
//*
```

After you have entered an appropriate job card, press **Enter**.

The generation process runs and presents you with a list of job streams and files that have been created to perform the migration to Version 6.1. These jobs do not require any editing; they are to be submitted "as-is" during the migration process.

17. When processing has completed, press **Enter** to proceed.

Example:

```
Processing for data set 'hlq.CNTL' ...
Member BBOWMG3D successfully created.
Member BBOMDHFS successfully created.
Member BBOMDCR successfully created.
Member BBOMDCRZ successfully created.
Member BBOMDDN successfully created.
Member BBOMDDNZ successfully created.
Member BBOMDSR successfully created.
Member BBOMDSRZ successfully created.
Member BBOMDCP successfully created.
Member BBOMDINS successfully created.
```

```
Processing for data set 'hlq.DATA' ...
Member BBOWBMPT successfully created.
Member BBOWMDRF successfully created.
***
```

18. Save your configuration variables for future reference.
 - a. On the **Deployment Manager Migration** menu, select option **S: Save customization variables**.
 - b. On the **Save Customization Variables** panel, specify the name of the data set into which to save your configuration variables and then click **Enter**.
 - c. On the **Allocate New Data Set** panel, which specifies the parameters for the .SAVECFG data set, click Enter to accept the defaults and return to the **Deployment Manager Migration** menu.

You have now completed the job generation process and are ready to begin the process of migration.

19. See the instructions for performing the migration by selecting option **4: View instructions**.

Also, see "Migrating a deployment manager" for instructions on performing the migration.

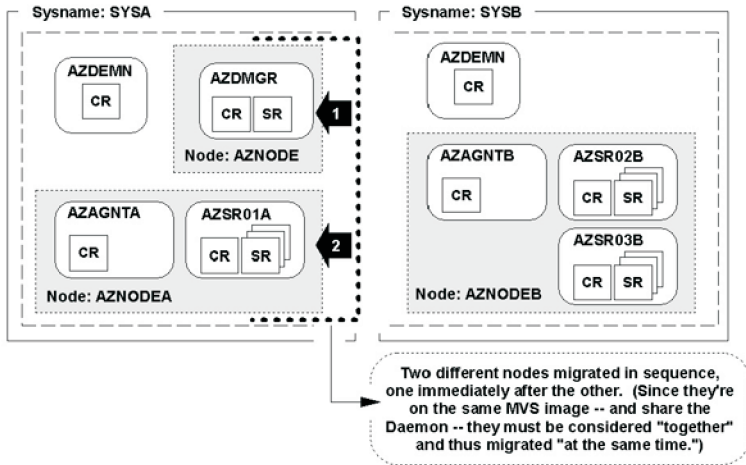
Migrating a deployment manager

After you use the Customization Dialog to generate JCL jobs for migrating a deployment manager to WebSphere Application Server for z/OS Version 6.1, you can perform the actual migration by running those jobs. For help in running these jobs, see the instructions that were generated when you generated the jobs.

See "Overview of migration, coexistence, and interoperability" on page 77 and "Premigration considerations" on page 81.

The deployment manager node is always migrated first. Review the applicable coexistence requirements for your system. If you are migrating from Version 5.0.2, you need to migrate other nodes in the same cell and on the same MVS image as the deployment manager sequentially, one immediately after the other.

The BBOWMG3D job referenced below must be submitted by a WebSphere Administrator User ID. All other jobs must be submitted by a user ID that has control over the file system.



Note:

Migrating a deployment manager from Version 5.x to Version 6.1 redeploys all application binaries when the deployment manager is restarted. This action results in the deployment manager recycling all the applications across the sysplex. This can cause an outage in a sysplex that is set up for high availability if the synchronization settings are not disabled.

To maintain high availability during the migration, turn off all synchronization options for all of the node agents in the sysplex before restarting the deployment manager.

1. Open the administrative console.
2. Go to **System administration > Node agents**.
3. For each node agent in the sysplex, go to **node_agent_server_name > File synchronization service** and disable all synchronization processing.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

Tip: Before migrating a WebSphere Application Server Version 5.x or 6.0.x deployment manager, use the **backupConfig** command or your own preferred backup utility to back up your existing configuration if you want to be able to restore it to its previous state after migration. Make sure that you note the exact name and location of this backed-up configuration.

1. Create and mount a new Version 6.1 configuration file system.

Before you perform the migration, Version 6.1 requires a configuration file system to be present for your new configuration. You can run BBOMDHFS or BBOMDZFS to create and mount a new configuration file system, or you can mount one manually. Either way, you must have a configuration file system for your Version 6.1 configuration created and mounted before you proceed. This configuration file system is the target of the migration; your Version 5.x or 6.0.x configuration file system is the source.

BBOMDHFS or BBOMDZFS creates a mount point directory, allocates the configuration’s file system, and mounts the file system at whatever value you specified on the **Mount point** field in the **System Environment Customization** panel in the Customization Dialog.

Ensure that you have allocated, created, and mounted your configuration file system data sets either manually or using BBOMDHFS or BBOMDZFS before you proceed. The mount point should be owned by the WebSphere Admin ID, and have permissions of at least 755. The new configuration file system structures should be included in BPXPARM so that they will be mounted at the next IPL.

2. Copy your generated JCL procedures.

The migration utility BBOMDCP copies the generated JCL procedures to start the servers to the specified procedure library. Your Version 6.1 configuration must use different JCL procedures from those used by your Version 5.x or 6.0.x configuration. This utility will update the new Version 6.1 configuration, substituting your new JCL names in place of the names that existed in your original Version 5.x or 6.0.x configuration.

Caution: This utility copies the dialog-generated JCL to your procedure library. If you specified the same names in the Customization Dialog as you used in your Version 5.x or 6.0.x configuration, this utility will overlay the existing procedures. If you are using the same names, make sure that you back up the existing Version 5.x or 6.0.x procedures before running this utility in case you need to roll back later.

Submit BBOMDCP, and verify a return code of 0.

3. If you specified new procedure names, update your RACF STARTED profiles for the controller and daemon.

The STARTED profile used by controller regions is based on the procedure name and JOBNAME. You must ensure that a STARTED profile will apply so that the proper identity will be assigned to the started task. For example, if your Version 5.x or 6.0.x deployment manager controller JCL procedure name is AZDCR, and you specified AZ1DCR for Version 6.1, then you would need to create a STARTED profile for that new procedure name:

new controller	same identity used in
JCL name	V5.x or 6.0.x configuration
<pre>RDEFINE STARTED AZ1DCR.* STDATA(USER(AZDCRU) GROUP(AZCFG) TRACE(YES))</pre>	

Note:

- Do not use a different user ID to start. There are other things tied to the user ID, and other changes would also be required if you change the user ID.
- If your original STARTED profile was generic, STARTED AZ*.* ... for example, you would not need to create a new STARTED profile.
- Servant region STARTED profiles are based on JOBNAME, not procedure name. So there is no issue with the servant when you use a different procedure name.
- Daemons and node agents are controllers, so using different procedure names for those implies a new STARTED profile.

4. Stop the Version 5.x or 6.0.x deployment manager.

You must stop the Version 5.x or 6.0.x deployment manager before submitting BBOWMG3D.

5. Submit BBOWMG3D.

A deployment manager migration does not require bringing the node into and out of PRR mode as stand-alone application server and federated node migrations do. There are two less jobs to submit for a deployment manager migration, therefore, and you are ready to perform the physical migration.

BBOWMG3D is the job that performs the physical migration of the Version 5.x or 6.0.x deployment manager to Version 6.1 based on the information you supplied in the Customization Dialog. Submit BBOWMG3D. Verify that you are getting return codes of 0, and review the log files in the migration temp directory on the configuration file system. The migration temporary directory is *temporary_directory_location/nnnnn*, where *temporary_directory_location* is the directory specified for **Temporary Directory Location** (/tmp/migrate by default) and *nnnnn* is the numeric value specified for **Migration identifier** on the **Server Customization (2 of 2)** panel in the Customization Dialog.

6. Shut down the old application servers and daemon.

Ensure that all nodes on the deployment manager's LPAR in the same cell have been shut down.

7. Update daemon JCL procedure if necessary.

WebSphere Application Server for z/OS Version 6.1 requires that the daemon process be at the highest level of code of any of the servers that it manages on the same LPAR. It will be at the Version 6.1 level when the deployment manager is started. If the deployment manager manages nodes on the

same LPAR in a mixed-cell environment, the daemon started JCL procedure for both the deployment manager and the down-level nodes must have both the Version 6.1 libraries and those of the highest level of the down-level nodes in STEPLIB.

If you are migrating from Version 5.1 and you have an application server node on the same LPAR as the deployment manager, for example, add the following to both your Version 6.1 and Version 5.1 daemon JCL procedure's "Z" member ("Z=BBO5DMNZ") using your library names:

```
//*STEPLIB Setup
//*
//STEPLIB DD DSN=hlq61.SBBOLD2,DISP=SHR
//        DD DSN=hlq61.SBBLOAD,DISP=SHR
//        DD DSN=hlq61.SBBOLPA,DISP=SHR
//        DD DSN=hlq51.SBBOLD2,DISP=SHR
//        DD DSN=hlq51.SBBOLPA,DISP=SHR
//*
```

Note: The *hlq51.SBBLOAD* library is intentionally left out of this example. It will not do any harm if it is there, but it is not needed.

After you migrate all nodes to Version 6.1 and before you remove the previous version's libraries from the system, you must update the daemon JCL procedure and remove the previous version's libraries from the STEPLIB concatenation. Failure to do so will result in a failure of the daemon to start.

For more information on STEPLIB, see the *Installing your application serving environment* PDF.

8. Start the new deployment manager.

Use the existing commands that you would use to start a Version 5.x or 6.0.x application server, but replace the RACF STARTED procedure name with the value you entered in the deployment manager **Server Customization (2 of 2)** panel for the **Controller procedure name** in the Customization Dialog. This command starts the Version 6.1 deployment manager. Wait until the server is finished initializing before proceeding.

The following message is displayed on the console and in the job log of BBODMGR:

```
BB000019I INITIALIZATION COMPLETE FOR WEBSHERE FOR z/OS CONTROL PROCESS BBODMGR
```

At this point, your migration to Version 6.1 is complete.

9. Perform the post-migration tasks.

After you have verified a successful migration to Version 6.1 and are successfully running a migrated configuration, you should delete the following items:

- Everything in the source configuration's file system
- Everything in the target configuration's *temporary_directory_location/nnnnn* directory, where *temporary_directory_location* is the directory specified for **Temporary Directory Location** (/tmp/migrate by default) and *nnnnn* is the numeric value specified for **Migration identifier** on the **Server Customization (2 of 2)** panel in the Customization Dialog
- The *bbomigt2.sh* file

Migrating federated nodes

The migration documentation for federated nodes includes a detailed WebSphere Application Server Version 6.1 migration explanation, a Customization Dialog walk-through, and a high-level migration checklist.

See "Overview of migration, coexistence, and interoperability" on page 77 and "Premigration considerations" on page 81.

Perform this task to migrate a WebSphere Application Server for z/OS Version 5.x or 6.0.x federated node to a Version 6.1 federated node.

Tip: For help in troubleshooting problems when migrating, see Chapter 11, "Troubleshooting migration," on page 149.

1. Print the “Checklist of activities for migrating a federated node,” and use it as you conduct your migration.
2. Inventory your applications and capture key information as described in “Preparing to migrate a Network Deployment configuration” on page 104.
3. Follow the instructions in “Customization Dialog walk-through for migrating a federated node” to create a set of migration jobs.
4. Follow the instructions in “Migrating a federated node” on page 123 to migrate the federated node.

Checklist of activities for migrating a federated node

Activities for migrating a federated node after you have generated the appropriate JCL jobs using the Customization Dialog include activities related to general preparation, transaction-log clearance, and node migration.

This checklist is intended to provide a quick reference.

Prepare to migrate the federated node

Check off	Item
	Make sure that you have a minimum level of Version 6.1 maintenance. See “Overview of migration, coexistence, and interoperability” on page 77.
	Create CNTL/DATA data sets for each node to be migrated. See “Customization Dialog walk-through for migrating a federated node”.
	Create Version 6.1 configuration mount point; allocate configuration file system, and mount either manually or using BBOMMHFS or BBOMMZFS.
	Run BBOMMCP to modify the JCL start procedure names.

Clear the transaction logs for the federated node

Check off	Item
	Stop the federated application server.
	Run BBOWMG1F, and check for success.
	Restart the federated application server to clear the transaction logs. It will start and then automatically stop.
	Run BBOWMG2F to disable PRR mode.

Migrate the federated node

Check off	Item
	Ensure that all servers are stopped, including the daemon server.
	Run BBOWMG3F, and check for success.
	Start the federated application server.

Customization Dialog walk-through for migrating a federated node

Walk through using the Customization Dialog to generate the JCL jobs (CNTL and DATA data sets) for migrating a federated application server node. You must create these jobs before you can begin the actual migration.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

Before migrating a federated node, you must migrate its deployment manager first. If you have not migrated the deployment manager, see “Migrating Network Deployment configurations” on page 104 then return to this article after the deployment manager is migrated.

Note: You can reuse the same target data sets, file system, and JCL procedures in the Customization Dialog walk-through for your federated nodes that you used for the deployment manager if you so choose. See the system-generated instructions for BBOMMINS for complete details.

Default values can be accepted for all fields and settings except for those that are explicitly identified in the following steps.

1. Invoke the WebSphere Application Server for z/OS Version 6.1 Customization Dialog.

ex 'product_hlq.SBBOCLIB(BBOWSTRT)' 'options'

- Example *product_hlq*: WAS610.WAS
- Example *options*: appl(az)

For more information on starting the Customization Dialog, including information on available options, see the *Installing your application serving environment* PDF.

2. From the first menu page, select option **4: Migrate a Node**.
3. On the next panel, choose option **3: Migrate a federated node**.
4. Allocate partitioned data sets that you will use to store the generated migration jobs and supporting data.
 - a. On the **Federated Node Migration** menu, choose option **1: Allocate target data sets**.
 - b. On the **Allocate Target Data Sets** panel, specify your high-level qualifier and then press **Enter** to proceed to the next panel.
 - c. Accept the defaults on the next two panels that specify the parameters for the .CNTL and .DATA data sets.

You are taken back to the **Federated Node Migration** menu.

5. On the **Federated Node Migration** menu, choose option **2: Define variables**.

In the following panels, the migration data collection process begins.

6. On the **Define Variables to Migrate a Federated Node** menu, choose option **1: System locations (directories, HLQs, etc.)**.
7. On the **System Locations (1 of 2)** panel, specify your Version 6.1 installation libraries and whether you intend to put your Version 6.1 load modules in STEPLIB.

Correctly specifying STEPLIB is essential to a successful migration. It is likely that your Version 5.x or 6.0.x modules are currently in LPA/LNKLST and that you will begin with your Version 6.1 libraries being defined in STEPLIB.

For more information on STEPLIB, see the *Installing your application serving environment* PDF.

Example:

System Locations (1 of 2)

Specify the following V6.1 information, then press ENTER to continue.

For some data sets, specify "Y" if they are in STEPLIB.

Full Names of Data Sets

PROCLIB.: SYS1.PROCLIB

Run WebSphere Application Server from STEPLIB (Y/N)? Y

```
SBBOLPA.: WAS610.WAS.SBBOLPA
SBBLOAD: WAS610.WAS.SBBLOAD
SBBGLOAD: WAS610.WAS.SBBGLOAD
SBBOLD2.: WAS610.WAS.SBBOLD2
```

Press **Enter** to proceed.

8. On the **System Locations (2 of 2)** panel, specify your Version 6.1 WebSphere Application Server product directory (/usr/lpp/zWebSphere/V6R1 for example) and press **Enter** to proceed.

Example:

System Locations (2 of 2)

Specify the following, then press Enter to continue.

```
V6.1 WebSphere Application Server product directory:
/usr/lpp/zWebSphere/V6R1
```

After specifying your Version 6.1 home directory, press **Enter** to proceed.

You are taken back to the **Define Variables to Migrate a Federated Node** menu, where option 1 is marked as completed.

9. On the **Define Variables to Migrate a Federated Node** menu, choose option **2: System environment customization**.
10. On the **System Environment Customization** panel, specify the Version 6.1 configuration file system. The configuration file system is where the configuration for the node being migrated is physically stored. You can choose to use an existing Version 6.1 file system if you already have an appropriate file system on the node being migrated. If you choose to use an existing Version 6.1 file system, you need to ensure that the mount point you specify here is present before running the migration utilities (BBOWMG1F, BBOWMG2F, and so on) that are created through these dialogs. If you choose to create a new Version 6.1 file system on the node being migrated, the actual creation of the new file system will not occur until you run the BBOMMHFS or BBOMMZFS job during the migration process after you complete this walk-through. See “Migrating a federated node” on page 123 for more information.

Example:

System Environment Customization

Specify the following to customize your system environment, then press Enter to continue.

WebSphere Application Server for z/OS configuration file system information

```
Mount point....: /WebSphere/V6R1
Name.....: OMVS.WAS.CONFIG.HFS
Volume, or '*' for SMS.: *
Primary allocation in cylinders...: 420
Secondary allocation in cylinders.: 100
```

File system type (HFS or ZFS): HFS

After specifying either an existing mount point or a new mount point and the configuration file system type, press **Enter** to proceed.

You are taken back to the **Define Variables to Migrate a Federated Node** menu, where options 1 and 2 are marked as completed.

11. On the **Define Variables to Migrate a Federated Node** menu, select option **3: Server customization**.
12. On the **Server Customization (1 of 2)** panel, specify or leave the defaults for all values.
 - a. Specify the source node that you are migrating under **From WebSphere Application Server home directory**.

- b. Specify the home location of the profile that will contain your Version 6.1 migrated node under **V6.1 WebSphere Application Server home directory**.
- c. Specify the procedure names used to start the Version 6.1 servers.
Your Version 6.1 configuration must use different JCL procedures from those used by your Version 5.x or 6.0.x configuration. The migration process creates new Version 6.1 JCL procedures using the procedure names specified here.
If you want to use your existing procedure names, specify "N" in the **Replace Started Procedure Command Names** field.

Caution: If you use the same names as you used in your Version 5.x or 6.0.x configuration, the migration process overlays the existing procedures. If you are using the same names, make sure that you back up the existing Version 5.x or 6.0.x procedures before running the jobs generated by the Customization Dialog in case you need to roll back later.

Example:

Server Customization (1 of 2)

Specify the following to customize your migration, then press Enter to continue.

From WebSphere Application Server home directory:
/WebSphere/V5R1M0
/ AppServer

V6.1 WebSphere Application Server home directory:
/WebSphere/V6R1
/ AppServer

Replace Started Procedure Command Names: Y

Daemon Procedure name.....: BB06DMN
Controller Procedure name.....: BB06ACR
Servant Procedure name.....: BB06ASR
Adjunct Procedure name.....: BB06CRA

After you have specified or left the defaults for all values, press **Enter** to proceed to the next panel.

13. On the **Server Customization (2 of 2)** panel, specify the migration options and application migration settings.

- a. Specify whether or not to migrate to support script compatibility.

This field specifies whether (Y) or not (N) migration should create the following Version 5.x or 6.0.x configuration definitions:

- Transport
- ProcessDef
- Version 5.x or 6.0.x SSL
- Version 5.x or 6.0.x ORB service threadpool

instead of the following Version 6.1 configuration definitions:

- Channels
- ProcessDefs
- Version 6.1 SSL
- Version 6.1 ORB service threadpool

Choose to migrate to support script compatibility if you want to minimize impacts to existing administration scripts. If you have existing **wsadmin** scripts or programs that use third-party

configuration APIs to create or modify the Version 5.x or 6.0.x configuration definitions, for example, you might want to choose this option during migration.

Note: This is meant to provide a temporary transition until all of the nodes in the environment are at the Version 6.1 level. When they are all at the Version 6.1 level, you should perform the following actions:

- 1) Modify your administration scripts to use all of the Version 6.1 settings.
- 2) Use the **convertScriptCompatibility** command to convert your configurations to match all of the Version 6.1 settings.

See “convertScriptCompatibility command” on page 129.

- b. Specify whether or not to enable tracing on the migration utilities.

If you choose to enable tracing, it will remain enabled throughout the entire migration process.

- c. Specify the location of the temporary directory for the migration configuration backup, temporary, output, and error files.

During migration, a backup copy of the Version 5.x or 6.0.x configuration is required. The default location of this backup is already specified, though you can override if needed. You might need to specify a location other than the default if the /tmp file system does not have adequate space to store the backup configuration. If you choose to override the default location of the backup copy, the best practice is to keep the same naming convention and just replace the /tmp portion with another path, /myTemp/migrate for example.

- d. Note the migration identifier.

The five-digit migration identifier (54700 in this example) is generated uniquely each time you create the migration jobs. The migration output messages, which you will need to monitor throughout the migration process, are stored in /tmp/migrate/54700. (After migration, the job output in this directory is NOT automatically deleted.) The migration output messages are also appended to the JCL sysout messages, which can be viewed in SDSF.

- e. Specify your preference for the migration of applications.

Choose one of the following options:

Y Install user enterprise applications in the Application Installation Directory as part of the migration.

S Prepare user enterprise applications for installation in the WebSphere Application Server Version 6.1 installableApps directory without actually installing them during migration.

JACL scripts that can be used to install these applications are generated and saved in the migration backup directory. You can then run these files at any point and in any combination after migration has completed. You can also reorganize and combine these JACL files for better applications installation efficiency if you want.

P Install user enterprise applications as part of the migration, and keep the same application installation directories as the previous version.

Restrictions: If you select this option, the location is shared by the existing WebSphere Application Server Version 5.x or 6.0.x installation and the Version 6.1 installation. If you keep the migrated applications in the same locations as those of the previous version, the following restrictions apply:

- The WebSphere Application Server Version 6.1 mixed-node support limitations must be followed. This means that the following support cannot be used when evoking the **wsadmin** command:
 - Precompile JSP
 - Use Binary Configuration
 - Deploy EJB

- You risk losing the migrated applications unintentionally if you later delete applications from these locations when administering (uninstalling for example) your Version 5.x or 6.0.x installation.

N Do nothing with user enterprise applications.

WebSphere Application Server system applications will migrate regardless of the value set here.

- Specify the directory in which to install the applications if you choose to install user enterprise applications in the Application Installation Directory as part of the migration .
- Specify an administrator user ID and password.

Because the migration jobs perform administrative functions, you need to supply a valid administrator user ID and password here.

Example:

Server Customization (2 of 2)

Specify the following to customize your migration, then press Enter to continue.

Migration Options

```
Migrate to support script compatibility: Y
Enable Tracing: Script: N Profile: N PreUpgrade: N PostUpgrade: N
Temporary Directory Location: /tmp/migrate
Migration Identifier.....: 54700
```

Application Migration Settings:

```
Application Migration Preference: Y (Y/S/P/N)
(Y) Migrate and install applications using the Application
    Installation Directory.
(S) Migrate Applications and generate scripts for later installation.
(P) Migrate Applications and keep the same application installation
    directories as the previous version.
(N) Do not migrate applications.
```

```
Application Installation Directory:
==> /WebSphere/V6R1/profiles/default/installedApps
```

```
WebSphere Administrator User ID.: XXXXXX
WebSphere Administrator Password: XXXXXXXX
```

Specify values or accept defaults, and press **Enter** to proceed. You are returned to the **Define Variables to Migrate a Federated Node** menu.

- Press **PF3** to return to the **Federated Node Migration** menu.
- On the **Federated Node Migration** menu, choose option **3: Generate customization jobs**.
- On the **Generate Customization Jobs** panel, provide a job card customized to your environment-specific requirements.

On the **Generate Customization Jobs** panel, you can see the names of the .CNTL and .DATA data sets that have been customized based on your previous input.

Example:

Generate Customization Jobs

This portion of the Customization Dialog generates the jobs you must run after you complete this dialog process. You must complete the customization process before you generate the jobs with this step. If you have not done this, please return to that step.

Jobs and data files will get generated into data sets:


```
'hlq.CNTL'  
'hlq.DATA'
```

If you wish to generate customization jobs using other data sets, then exit from this panel and select the "Allocate target data sets" option.

All the jobs that will be tailored for you will need a job card. Please enter a valid job card for your installation below. The file tailoring process will update the job name for you in all the generated jobs, so you need not be concerned with that portion of the job cards below. If continuations are needed, replace the comment cards with continuations.

Specify the job cards, then press Enter to continue.

```
//jobname JOB (ACCTNO,ROOM),'IBMUSER',CLASS=A,REGION=0M  
//*  
//*  
//*
```

After you have entered an appropriate job card, press **Enter**.

The generation process runs and presents you with a list of job streams and files that have been created to perform the migration to Version 6.1. These jobs do not require any editing; they are to be submitted "as-is" during the migration process.

17. When processing has completed, press **Enter** to proceed.

Example:

```
Processing for data set 'hlq.CNTL' ...  
Member BBOWMG1F successfully created.  
Member BBOWMG2F successfully created.  
Member BBOWMG3F successfully created.  
Member BBOMMHFS successfully created.  
Member BBOMMCR successfully created.  
Member BBOMMCRZ successfully created.  
Member BBOMMDN successfully created.  
Member BBOMMDNZ successfully created.  
Member BBOMMSR successfully created.  
Member BBOMMSRZ successfully created.  
Member BBOMMCP successfully created.  
Member BBOMMINS successfully created.
```

```
Processing for data set 'hlq.DATA' ...  
Member BBOWBMPT successfully created.  
Member BBOWMMRF successfully created.  
***
```

18. On the **Federated Node Migration** menu, save your configuration variables for future reference by selecting option **S: Save customization variables**.
You have now completed the job generation process and are ready to begin the process of migration.
19. See the instructions for performing the migration by selecting option **4: View instructions**.
Also, see "Migrating a federated node" for additional instructions on performing the migration.

Migrating a federated node

After the deployment manager is migrated and restarted, you can perform the actual migration of its federated application server nodes by running the JCL jobs that you generated using the Customization Dialog. For help in running these jobs, see the instructions that were generated when you generated the jobs.

See "Overview of migration, coexistence, and interoperability" on page 77 and "Premigration considerations" on page 81.

Make sure that you go through the steps outlined in “Customization Dialog walk-through for migrating a federated node” on page 117 before you begin this migration. You will not be able to proceed if you have not created the JCL migration utilities through the Customization Dialog.

The BBOWMG1F, BBOWMG2F, and BBOWMG3F jobs referenced below must be submitted by a WebSphere Administrator user ID. All other jobs must be submitted by a user ID that has control over the file system.

To migrate a federated node on another MVS image, make sure that you run the jobs on the same system on which the node itself resides.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

Tip: When migrating a WebSphere Application Server Version 5.x or 6.0.x federated node, you must perform the following actions if you want to be able to roll it back to its previous state after migration:

1. Back up your existing configuration using the **backupConfig** command or your own preferred backup utility.
 - Run the **backupConfig** command or your own preferred utility to back up the Version 6.1 deployment manager configuration.

Important: Make sure that you note the exact name and location of this backed-up configuration.

- Run the **backupConfig** command or your own preferred utility to back up the Version 5.x or 6.0.x federated node configuration.

Important: Make sure that you note the exact name and location of this backed-up configuration.

2. Migrate the federated node.

If necessary, you can now roll back the federated node that you just migrated. See “Rolling back a managed node” on page 132.

1. Ensure that the application servers and the node agent are stopped for the federated node being migrated.

You must stop the Version 5.x or 6.0.x federated node before proceeding.

2. Ensure that the newly migrated deployment manager is up and running.

In order for the application server node to be properly migrated, the deployment manager must be running. In order for this migration to work, the deployment manager must be up and listening on its SOAP port.

Check Off	Item
	Access the administrative console of the WebSphere Application Server Version 6.1 deployment manager. This validates that the deployment manager is running.

Check Off	Item
	Ensure that the WebSphere Application Server Version 6.1 copy of the code is running. Under “About your WebSphere Application Server,” the build number should begin with W61.

3. Create and mount a new Version 6.1 configuration file system.

Before you perform the migration, Version 6.1 requires that a configuration file system is present for your new configuration. You can run BBOMMHFS or BBOMMZFS to create and mount a new configuration file system, or you can mount one manually. Either way, you must have a file system for

your Version 6.1 configuration created and mounted before you proceed. This configuration file system is the target of the migration; your Version 5.x or 6.0.x configuration file system is the source. BBOMMHFS or BBOMMZFS creates a mount-point directory, allocates the configuration's file system, and mounts the file system at whatever value you specified on the **Mount point** field in the **System Environment Customization** panel of the Customization Dialog.

Before you proceed, ensure that you have allocated, created, and mounted your configuration file system data sets either manually or using BBOMMHFS or BBOMMZFS. The mount point should be owned by the WebSphere Administrator user ID and have permissions of at least 755. The new configuration file system structures should be included in BPXPARM so that they will be mounted at the next IPL.

4. Copy your generated JCL procedures.

The migration utility BBOMMCP copies the generated JCL procedures to start the servers to the specified procedure library. Your Version 6.1 configuration must use different JCL procedures from those used by your Version 5.x or 6.0.x configuration. This utility will update the new Version 6.1 configuration, substituting your new JCL names in place of the names that existed in your original Version 5.x or 6.0.x configuration.

Caution: This utility copies the dialog-generated JCL to your procedure library. If you specified the same names in the Customization Dialog as you used in your Version 5.x or 6.0.x configuration, this utility will overlay the existing procedures. If you are using the same names, make sure that you back up the existing Version 5.x or 6.0.x procedures before running this utility in case you need to roll back later.

Submit BBOMMCP, and verify a return code of 0.

5. If you specified new procedure names, update your RACF STARTED profiles for the controller and daemon.

The STARTED profile used by controller regions is based on the procedure name and JOBNAME. You must ensure that a STARTED profile will apply so that the proper identity will be assigned to the started task. If your Version 5.x or 6.0.x controller JCL procedure name is AZACR and you specified AZ1ACR for Version 6.1, for example, then you would need to create a STARTED profile for that new procedure name:

new controller JCL name	same identity used in V5.x or 6.0.x configuration
<pre>RDEFINE STARTED AZ1ACR.* STDATA(USER(AZACRU) GROUP(AZCFG) TRACE(YES))</pre>	

Note:

- Do not use a different user ID to start. There are other things tied to the user ID, and other changes would also be required if you change the user ID.
- If your original STARTED profile was generic, STARTED AZ*.* ... for example, you would not need to create a new STARTED profile.
- Servant region STARTED profiles are based on JOBNAME, not procedure name. So there is no issue with the servant when you use a different procedure name.
- Daemons and node agents are controllers; so, using different procedure names for these implies a new STARTED profile.

6. Submit BBOWMG1F and BBOWMG2F.

Note: If you are not using XA connectors, submitting BBOWMG1F and BBOWMG2F is optional. However, you should submit both jobs to ensure that your transaction logs are clear.

BBOWMG1F enables all servers on the federated application server node being migrated to start in PRR processing mode. PRR processing mode resolves any outstanding transactions, clears the transaction logs, and terminates the server. BBOWMG2F disables PRR mode and returns all servers to normal operating state.

Follow these steps to clear the XA transaction logs:

- a. Submit the job BBOWMG1F, and verify a return code of 0.
 - b. Restart the federated application server, and wait for it to perform PRR processing and terminate automatically.
 - c. Submit the job BBOWMG2F, and verify a return code of 0.
7. Delete and redefine the log stream if necessary.

Perform this step only if you previously configured the transaction XA partner log or compensation log on the Version 5.x or 6.0.x server to use a log stream.

- a. Make sure that the node is shut down.
- b. Delete and redefine the log stream.

You can use the BBOLOGSD and BBOLOGSA jobs that were created during Version 5.x or 6.0.x customization if you configured the server initially to use the log stream.

The following sample shows an example of such a job:

```
//RLSP1A JOB 'xxxx,yyy,?', 'USERID',MSGCLASS=H,
//          CLASS=J,MSGLEVEL=(1,1),REGION=4M,NOTIFY=&SYSUID
//STEP1    EXEC PGM=IXCMIAPU
//STEPLIB DD DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *

DATA TYPE(LOGR) REPORT(YES) /* Default to show output of job */
DELETE LOGSTREAM NAME(P1ACEL6A.W51ASA2.D)
DEFINE LOGSTREAM NAME(P1ACEL6A.W51ASA2.D)
        LOWOFFLOAD(20)
        HIGHOFFLOAD(79)
        STG_DUPLEX(YES)
        DUPLEXMODE(UNCOND)
        STG_DATACLAS(OPERLOG)
        STG_SIZE(5000)
        HLQ(Q10RRS)
        LS_SIZE(5000)
        LS_DATACLAS(OPERLOG)
        STRUCTNAME(WAS_LOGRLS)

/*
```

If you are migrating nodes in a sysplex, follow this procedure for each federated node that you migrate.

8. Submit BBOWMG3F.

BBOWMG3F is the job that performs the physical migration of the Version 5.x or 6.0.x node to Version 6.1 based on the information you supplied in the Customization Dialog. Submit BBOWMG3F, verify that you are getting return codes of 0, and review the log files in the temporary migration directory on the configuration file system. The migration temporary directory is *temporary_directory_location/nnnnn*, where *temporary_directory_location* is the directory specified for **Temporary Directory Location** (/tmp/migrate by default) and *nnnnn* is the numeric value specified for **Migration identifier** on the **Server Customization (2 of 2)** panel in the Customization Dialog.

9. Ensure that the old daemon is shut down.

Ensure that all federated nodes in the same cell on this LPAR have been shut down.

10. Update daemon JCL procedure if necessary.

WebSphere Application Server for z/OS Version 6.1 requires that the daemon process be at the highest level of code of any of the servers that it manages on the same LPAR. It will be at the Version 6.1 level when this federated node is started. If there are any additional nodes on the same LPAR and in the same cell as this migrated managed node, you must update the additional down-level node's daemon JCL procedure to STEPLIB to both the Version 6.1 libraries and those of the highest-level of the down-level nodes.

If you are migrating from Version 5.1 and you have an additional application server node on the same LPAR and in the same cell, for example, add the following to both your Version 6.1 and Version 5.1 daemon JCL procedure's "Z" member ("Z=BBO5DMNZ") using your library names:

```

//*STEPLIB Setup
//*
//STEPLIB DD DSN=hlq61.SBBOLD2,DISP=SHR
//        DD DSN=hlq61.SBBOLOAD,DISP=SHR
//        DD DSN=hlq61.SBBOLPA,DISP=SHR
//        DD DSN=hlq51.SBBOLD2,DISP=SHR
//        DD DSN=hlq51.SBBOLPA,DISP=SHR
//

```

Note: The *hlq51.SBBOLOAD* library is intentionally left out of this example. It will not do any harm if it is there, but it is not needed.

After you migrate all nodes to Version 6.1 and before you remove the previous version's libraries from the system, you must update the daemon JCL procedure and remove the previous version's libraries from the STEPLIB concatenation. Failure to do so will result in a failure of the daemon to start.

See the *Installing your application serving environment* PDF for more information on STEPLIB.

11. Start the new federated application server node.

a. Start the node agent.

The following message is displayed on the console and in the job log of BBON001:

```
BB000019I INITIALIZATION COMPLETE FOR WEBSphere FOR z/OS CONTROL PROCESS BBON001
```

b. Start the federated application server.

Use the existing command that you would use to start a Version 5.x or 6.0.x application server, but replace the RACF STARTED procedure name with the value that you entered in the federated node **Server Customization (2 of 2)** panel for the **Controller procedure name** in the Customization Dialog. This command starts the Version 6.1 federated application server. Wait until the server is finished initializing before proceeding.

The following message is displayed on the console and in the job log of BBOS001:

```
BB000019I INITIALIZATION COMPLETE FOR WEBSphere FOR z/OS CONTROL PROCESS BBOS001
```

At this point, your migration to Version 6.1 is complete.

12. Perform the post-migration tasks.

After you have verified a successful migration to Version 6.1 and are successfully running a migrated configuration, you should delete the following items:

- Everything in the source configuration's file system
- Everything in the target configuration's *temporary_directory_location/nnnnn* directory, where *temporary_directory_location* is the directory specified for **Temporary Directory Location** (/tmp/migrate by default) and *nnnnn* is the numeric value specified for **Migration identifier** on the **Server Customization (2 of 2)** panel in the Customization Dialog
- The *bbomigr2.sh* file

Using the migration tools to migrate product configurations

Migration support consists of tools that are shipped as part of WebSphere Application Server Version 6.1 that primarily provide support for saving the configuration and applications from an older version of WebSphere into a migration-specific backup directory and then importing that configuration into Version 6.1.

See "Overview of migration, coexistence, and interoperability" on page 77 and "Premigration considerations" on page 81.

It is important to use the migration tools for the version of WebSphere Application Server for z/OS that you are installing. The tools change over time. If you use migration tools from an earlier release of WebSphere Application Server for z/OS, you are likely to encounter a problem with the migration.

Select the appropriate migration tools to migrate your product configurations.

Customization Dialog

Used to collect the information and generate the JCL jobs (CNTL and DATA data sets) for migrating a stand-alone application server, deployment manager, or federated node. You must create these jobs before you can begin the actual migration.

The Customization Dialog, is run from TSO using the following to launch it:

```
ex 'product_hlq.SBBOCLIB(BBOWSTR)' 'options'
```

Commands

clientUpgrade

Upgrades the client application to a new release level.

The **clientUpgrade** tool is in the WAS_HOME/bin directory after installation.

convertScriptCompatibility

Used by administrators to convert their configuration from a mode that supports backward compatibility of Version 5.x or 6.0.x administration scripts to a mode that is fully Version 6.1.

The **convertScriptCompatibility** tool is in the WAS_HOME/bin directory after installation.

See “convertScriptCompatibility command” on page 129.

Tip: For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

Use the selected tools to migrate your product configuration.

The clientUpgrade command

Use the **clientUpgrade** command to migrate previous versions of client resources to Version 6.1 level resources.

Use the **clientUpgrade** command to migrate Version 5.x client resources to Version 6.1 level resources. In the process of migrating these resources, the client-resources.xmi file located in the client jars is migrated to the latest level. A backup of the client-resources.xmi file is also located in the client jar. If this command is not executed against the client EAR files before they are installed on Version 6.1, the client EARs will not operate or install correctly.

The command file is located in the bin subdirectory of the WAS_HOME directory.

```
clientUpgrade 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
```

convertScriptCompatibility command

The **convertScriptCompatibility** command is used by administrators to convert their configurations from a mode that supports backward compatibility of WebSphere Application Server Version 5.x or 6.0.x administration scripts to a mode that is fully in the Version 6.1 configuration model.

The scope of the configuration changes depend on the type of profile that is being processed.

- For Express and base configurations, the default is to convert all servers owned by the node in that configuration.

Use the `-serverName` parameter for more granular control.

- For Network Deployment configurations, the default behavior is to convert all nodes and all servers owned by those nodes.

Use the `-nodeName` and `-serverName` parameters for more granular control.

Nodes are checked to verify that they are at a WebSphere Application Server Version 6.1 level before they are processed in order to support mixed-node configurations. Client environments are not processed.

The following conversions take place with this tool:

- `processDef` to `processDefs`

WCCM objects of type `processDef` from WebSphere Application Server Version 5.x are converted to use `processDefs` as defined in the Version 6.1 `server.xml` model. The existing `processDef` object remains in the configuration and is ignored by the runtime.

- `transports` to `channels`

Existing transport entries in the configuration from WebSphere Application Server Version 5.x are mapped to channel support. This affects `server.xml` and `serverindex.xml` files. The values of the transport settings are used to create new channel entries.

- SSL configuration

WebSphere Application Server Version 6.1 contains enhancements to SSL configuration that result in refactoring the existing SSL configuration model. Both the old and the new model are supported. The default is to map to the WebSphere Application Server Version 5.x or 6.0.x SSL configuration model.

Syntax

The syntax is as follows:

```
convertScriptCompatibility.sh [-help]
```

```
convertScriptCompatibility.sh [-profileName profile_name]  
                             [-backupConfig true | false]  
                             [-nodeName node_name [-serverName server_name]]  
                             [-traceString trace_spec [-traceFile file_name]]
```

Parameters

Supported arguments include the following parameters:

-help

This displays help for this command

-backupConfig

This is an optional parameter that is used to back up the existing configuration of the current profile. The default is true—that is, 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

This is an optional parameter that is used to specify the profile configuration in the Version 6.1 environment. If this is not specified, the default profile is used. If the default profile has not been set or cannot be found, the system returns an error.

-nodeName

This is an optional parameter that is used to specify a particular node name be processed rather than every node in the configuration. If this is not specified, all nodes in the configuration are converted.

-serverName

This is an optional parameter that is used to specify a particular server name to be processed rather than every server in the configuration. It can be used on all profile types and can be used in conjunction with the *-nodeName* parameter when processing Network Deployment configurations. If this parameter is not specified, all servers in the configuration are converted. If it is used in conjunction with the *-nodeName* parameter, all processing is limited to the specified node name.

-traceString

This is an optional parameter. The value *trace_spec* specifies the trace information that you want to collect. To gather all trace information, specify "**=all=enabled*" (with quotation marks). The default is to not gather trace information. If you specify this parameter, you must also specify the *-traceFile* parameter.

-traceFile

This is an optional parameter. The value *file_name* specifies the name of the output file for trace information. If you specify the *-traceString* parameter but do not specify the *-traceFile* parameter, the command does not generate a trace file.

Migrating Cloudscape databases

After you use the migration tools to migrate to WebSphere Application Server Version 6.1, you should verify the results of the automatic Cloudscape database migration and manually migrate any Cloudscape database instances that are not automatically migrated by the tools.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

Tip: Before you run the migration tools, ensure that the *debug migration trace* is active. By default, this trace function is enabled. To reactivate the debug migration trace if it is disabled, set one of the following trace options:

- `all traces*=all`
- `com.ibm.ws.migration.WASUpgrade=all`

WebSphere Application Server Version 6.1 requires Cloudscape Version 10.1.

Cloudscape Version 10.1 is a pure Java database server that combines the Apache Derby runtime with the opportunity to use the full services of IBM Software Support. For comprehensive information about Cloudscape Version 10.1, see the Cloudscape section of [ibm.com](http://www.ibm.com/software/data/cloudscape/) at <http://www.ibm.com/software/data/cloudscape/>.

For help in troubleshooting problems when migrating, see Chapter 11, “Troubleshooting migration,” on page 149.

1. Verify the automatic migration of Cloudscape database instances.
When you migrate from WebSphere Application Server Version 5.x or 6.0.x to Version 6.1, the migration tools automatically upgrade the database instances that are accessed through the embedded framework by some internal components such as the UDDI registry. The tools also attempt to upgrade Cloudscape instances that your applications access through the embedded framework. You must verify these migration results after running the migration tools.
See “Verifying the Cloudscape v10.1.x automatic migration” on page 50.
2. Manually migrate Cloudscape database instances where necessary.
The Version 6.1 migration tools do not attempt to automatically migrate database instances that transact with applications through the Cloudscape Network Server framework. This exclusion eliminates the risk of corrupting third-party applications that access the same database instances as those accessed by WebSphere Application Server.
For details on manually migrating database instances that are accessed through the Cloudscape Network Server framework as well as Cloudscape instances that fail the automatic migration, see “Upgrading Cloudscape manually” on page 53.
3. Manually migrate your UDDI registry if it uses a database on the Cloudscape Network Server framework.
See “Migrating the UDDI registry” on page 43.

Rolling back your environment

After migrating to a WebSphere Application Server Version 6.1 environment, you can roll back to a Version 5.x or 6.0.x environment. This returns the configuration to the state that it was in before migration. After rolling back the environment, you can restart the migration process.

Generally, migration does not modify anything in the configuration of the prior release; however, there are cases where minimal changes are made that are reversible—those of a deployment manager and its managed nodes.

- To roll back a Version 6.1 deployment manager and its managed nodes to Version 5.x or 6.0.x, follow the instructions in “Rolling back a deployment cell.”
- To roll back a Version 6.1 managed node to Version 5.x or 6.0.x, follow the instructions in “Rolling back a managed node” on page 132.

The configuration should now be returned to the state that it was in before migration.

You can now restart the migration process if you want to do so.

Rolling back a deployment cell

You can use the **restoreConfig** and **wsadmin** commands to roll back a migrated WebSphere Application Server Version 6.1 deployment cell to Version 5.x or 6.0.x. This returns the configuration to the state that it was in before migration. After rolling back the deployment cell, you can restart the migration process.

When migrating a WebSphere Application Server Version 5.x or 6.0.x deployment cell, you must perform the following actions if you want to be able to roll it back to its previous state after migration:

1. Back up your existing configuration using the **backupConfig** command or your own preferred backup utility.
 - Run the **backupConfig** command or your own preferred utility to back up the Version 5.x or 6.0.x deployment manager configuration.

Important: Make sure that you note the exact name and location of this backed-up configuration.

 - Run the **backupConfig** command or your own preferred utility to back up the Version 5.x or 6.0.x managed node configurations.

Important: Make sure that you note the exact name and location of these backed-up configurations.

2. Migrate the deployment cell.

If necessary, you can now roll back the deployment cell that you migrated.

1. Stop all of the servers that are currently running in the WebSphere Application Server Version 6.1 environment.
2. If you chose to disable the previous deployment manager when you migrated to the WebSphere Application Server Version 6.1 deployment manager, run the **restoreConfig** command or your own preferred utility to restore the Version 5.x or 6.0.x configuration for the deployment manager.

Important: Make sure that you restore the same backed-up configuration that you created just before you migrated the deployment manager.

3. Run the **restoreConfig** command or your own preferred utility to restore the WebSphere Application Server Version 5.x or 6.0.x configurations for the deployment cell's managed nodes that you need to roll back.

Important: Make sure that you restore the same backed-up configuration that you created just before you migrated each managed node.

4. If you chose to disable the previous deployment manager when you migrated to the WebSphere Application Server Version 6.1 deployment manager, use the **wsadmin** command to run the migrationDisablementReversal.jacl script from the Version 5.x or 6.0.x WAS_HOME/bin directory of the deployment manager that you need to roll back from Version 6.1.

Use the following parameters:

```
./wsadmin.sh -f migrationDisablementReversal.jacl -conntype NONE
```

5. If you chose to keep the installed applications in the same location as the prior release during migration to WebSphere Application Server Version 6.1 and any of the Version 6.1 applications are not compatible with the prior release, install applications that are compatible.
6. Start the rolled-back deployment manager and its managed nodes in the WebSphere Application Server Version 5.x or 6.0.x environment.

The configuration should now be returned to the state that it was in before migration.

You can now restart the migration process if you want to do so.

Rolling back a managed node

You can use the **restoreConfig** and **wsadmin** commands to roll back a migrated WebSphere Application Server Version 6.1 managed node to the state that it was in before migration. For each managed node that you want to roll back, you must roll back the managed node itself and the corresponding changes made to the master repository located on the deployment manager.

When migrating a WebSphere Application Server Version 5.x or 6.0.x managed node, you must perform the following actions if you want to be able to roll it back to its previous state after migration:

1. Back up your existing configuration using the **backupConfig** command or your own preferred backup utility.
 - Run the **backupConfig** command or your own preferred utility to back up the Version 6.1 deployment manager configuration.

Important: Make sure that you note the exact name and location of this backed-up configuration.

- Run the **backupConfig** command or your own preferred utility to back up the Version 5.x or 6.0.x managed node configuration.

Important: Make sure that you note the exact name and location of this backed-up configuration.

2. Migrate the managed node.

If necessary, you can now roll back the managed node that you just migrated.

You must perform all of the backup and rollback actions for each migrated managed node before you proceed to migrate another managed node.

1. Stop all of the servers that are currently running in the WebSphere Application Server Version 6.1 environment.
2. Restore your previous configuration using the **restoreConfig** command or your own preferred restoration utility.
 - a. Run the **restoreConfig** command or your own preferred utility to restore the Version 6.1 deployment manager configuration.

Important: Make sure that you restore the same backed-up configuration that you created just before you migrated the managed node.

- b. Run the **restoreConfig** command or your own preferred utility to restore the Version 5.x or 6.0.x configuration for the managed node that you need to roll back from Version 6.1.

Important: Make sure that you restore the same backed-up configuration that you created just before you migrated this managed node.

3. Use the **wsadmin** command to run the migrationDisablementReversal.jacl script from the WebSphere Application Server Version 5.x or 6.0.x WAS_HOME/bin directory of the managed node that you need to roll back from Version 6.1.

Use the following parameters:

```
./wsadmin.sh -f migrationDisablementReversal.jacl -conntype NONE
```

4. If you chose to keep the installed applications in the same location as the prior release during migration to WebSphere Application Server Version 6.1 and any of the Version 6.1 applications are not compatible with the prior release, install applications that are compatible.
5. Start the rolled-back managed node in the WebSphere Application Server Version 6.1 environment.

The configuration should now be returned to the state that it was in before migration.

You can now restart the migration process if you want to do so.

Chapter 6. Migrating Web server configurations

You can migrate a Web server from supporting an earlier version of WebSphere Application Server to support the current version.

Note on the applicability of the information in this article: This information in this article is only applicable if you migrated from WebSphere Application Server Version 5.x; it is not applicable if you migrated from Version 6.0.x.

The plug-in configuration file (plugin-cfg.xml) generated after successful migration from Version 5.x to Version 6.1 is topology centric—that is, it includes all the applications within a cell. You can manage this cell-wide plug-in configuration file from the Version 6.1 administrative console, by using the GenPluginCfg command, or by using the Plug-in Config Generator MBean.

Be aware that regenerating the plug-in configuration can overwrite manual configuration changes that you might want to preserve.

The application-centric generation of the plugin-cfg.xml file is supported using the Version 6.x administrative console. Being application centric means that the plugin-cfg.xml file generated in the administration console has a granularity that allows each application to be mapped to its specific Web or application server.

To set up the administrative console so that you can use it to manage the Web server plug-in configuration, you must first create a default Web server configuration and then use the administrative console to add the plug-in properties from your migrated plugin-cfg.xml file to this Web server configuration.

1. Create a default Web server configuration.
2. Add the plug-in properties from your migrated plugin-cfg.xml file to this Web server configuration.

Chapter 7. Migrating administrative scripts

You can migrate administrative scripts using scripting and the wsadmin tool.

WebSphere Application Server Version 6.1 supports migrating administrative scripts from V5.x.

- If you are migrating administrative scripts from V5.x, see:
“Migrating administrative scripts from 5.x to 6.x”
- If you are migrating administrative scripts from V6.0.2, see:
“Migrating administrative scripts from V6.0.2 to V6.1” on page 139

Migrating administrative scripts from 5.x to 6.x

There are some changes you should be aware of when migrating from WebSphere Application Server V5.x to V6.x

There are a few changes to be aware of that are required for your existing scripts when moving to WebSphere Application Server Version 6.x. In general, the administration model has changed very little. However, there are some changes required with Version 6.x. See the following steps for the three categories of changes:

Prepare for evolutionary changes **without** script compatibility support. These types of changes can be evolved directly without the assistance of script compatibility support. Read “Making changes from 5.x to 6.x to accommodate changes without scripting support” for more information.

Making changes from 5.x to 6.x to accommodate changes without scripting support

There are some changes without scripting support you should be aware of when migrating from WebSphere Application Server V5.x to V6.x, as they can affect script compatibility.

There are a few changes to be aware of that are required for your existing scripts when moving to WebSphere Application Server Version 6.x. 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. See the following steps for changes:

Prepare for evolutionary changes **without** script compatibility support.

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

```

Example: Migrating - Allowing configuration overwrite when saving a configuration

These examples demonstrate how to enable configuration overwrite in network deployment for WebSphere Application Server V5.x and V6.x.

Use the following examples:

- 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
```

On z/OS platforms:

```
./stopManager.sh  
./startManager.sh
```

3. Allow configuration overwrite, for example:

Using Jacl:

```
$AdminConfig setSaveMode overwriteOnConflict
```

Using Jython:

```
AdminConfig.setSaveMode('overwriteOnConflict')
```

Migrating administrative scripts from V6.0.2 to V6.1

There are some changes you should be aware of when migrating from WebSphere Application Server V6.0.2 to V6.1

There are a few changes to be aware of that are required for your existing scripts when moving to WebSphere Application Server Version 6.1. In general, the administration model has changed very little. However, there are some changes required with Version 6.1. See the following steps for the two categories of changes:

- Be aware of removed features that may have an impact on administration scripts. For more information, see the “Deprecated and removed features” on page 1 article. These may include the following:
 - Support for the Secure Authentication Service (SAS) IIOp security protocol.
 - Support for the Common Connector Framework (CCF).
 - Support for the IBM Cloudscape Version 5.1.x database.
 - Use of JSSE instead of System SSL for all servers except the Daemon.
 - DB2 for zOS Local JDBC Provider (RRS). For more information, see the “Premigration considerations” on page 81 article.
- Be aware of the change required when creating Service Integration Bus objects. For more information about the **createSIBus** command, see the Creating a service integration bus through the command line article.

Related tasks

Chapter 7, “Migrating administrative scripts,” on page 137

You can migrate administrative scripts using scripting and the `wsadmin` tool.

Chapter 8. Coexisting

You can create an environment in which multiple versions of WebSphere Application Server can run independently on the same system at the same time. A major consideration in coexistence is the avoidance of port conflicts.

Coexisting, as it applies to WebSphere Application Server products, involves multiple installations of WebSphere Application Server running independently on the same system at the same time. Multiple installations include multiple versions and multiple installations of one version. Coexistence also implies various combinations of Web server interaction.

See “Coexistence support.”

This article discusses which coexistence scenarios are supported.

Coexistence support

Coexistence is a state in which multiple installations and multiple nodes from different versions of WebSphere Application Server run independently in the same environment at the same time.

As it applies to WebSphere Application Server for z/OS products, coexistence primarily refers to the ability of multiple customizations (nodes) of WebSphere Application Server to run on the same z/OS image or sysplex at the same time. You can install WebSphere Application Server for z/OS once and customize it many times; that is, you can create many nodes from one installation. This installation can be shared by all LPARs (logical partitions) in the same sysplex.

When setting up your system for coexistence, you need to be aware of the points in the following table

Table 3. WebSphere Application Server for z/OS Version 5.x, Version 6.0.x, and Version 6.1 multiversion coexistence scenarios

Coexistence Type	Comments
General	<p>Multiple WebSphere Application Server for z/OS cells can coexist in the same sysplex.</p> <p>Multiple WebSphere Application Server for z/OS nodes can coexist on the same LPAR.</p> <p>WebSphere Application Server for z/OS Version 5.x, Version 6.0.x, and Version 6.1 nodes can all coexist in the same cell.</p> <p>No two cells can have the same cell short name.</p> <p>Separate cells need separate configuration file system mount points and JCL procedures.</p> <p>If your Version 5.x or 6.0.x load modules reside in LPA/LNKLST, you will need to place the Version 6.1 modules in STEPLIB in order to coexist. Only one release of WebSphere Application Server for z/OS can reside in LPA/LNKLST.</p> <p>You need to determine port conflicts that might occur when earlier versions coexist with Version 6.1.</p>

Table 3. WebSphere Application Server for z/OS Version 5.x, Version 6.0.x, and Version 6.1 multiversion coexistence scenarios (continued)

Coexistence Type	Comments
WebSphere Application Server for z/OS Version 5.0.x and Version 5.1	<p>Nodes can coexist in the same cell if and only if they are on different LPARs.</p> <p>Nodes can coexist on the same LPAR if and only if they are in different cells.</p>
WebSphere Application Server for z/OS Version 5.0.x and Version 6.0.x	<p>Nodes can coexist in the same cell if and only if they are on different LPARs. Note: A service level of at least W502025 is required for Version 5.0.x nodes to coexist with Version 6.0.x in the same cell.</p> <p>Nodes can coexist on the same LPAR if and only if they are in different cells.</p>
WebSphere Application Server for z/OS Version 5.0.x and Version 6.1	<p>Nodes can coexist in the same cell if and only if they are on different LPARs. Note: A service level of at least W502025 is required for Version 5.0.x nodes to coexist with Version 6.1 in the same cell.</p> <p>Nodes can coexist on the same LPAR if and only if they are in different cells.</p>
WebSphere Application Server for z/OS Version 5.1 and Version 6.0.x	<p>Nodes can coexist on the same LPAR in the same cell or in different cells. Note: A service level of at least W510207 is required for Version 5.1 nodes to coexist with Version 6.0.x in the same cell.</p>
WebSphere Application Server for z/OS Version 5.1 and Version 6.1	<p>Nodes can coexist on the same LPAR in the same cell or in different cells. Note: A service level of at least W510207 is required for Version 5.1 nodes to coexist with Version 6.1 in the same cell.</p>
WebSphere Application Server for z/OS Version 6.0.x and Version 6.1	<p>Nodes can coexist on the same LPAR in the same cell or in different cells.</p>

Restrictions on using mixed-release cells:

- You can upgrade a portion of the nodes in a cell to WebSphere Application Server Version 6.1 while leaving others at the older release level. This means that, for a period of time, you might be administering servers that are at the current release and servers that are running the newer release in the same cell.

Note that in this mixed-release environment, there might be some restrictions on what you can do with servers at the older release level. There might also be restrictions on creating clusters and cluster members. For details, see the information center.
- WebSphere Application Server Version 5.0.x cannot coexist with Version 6.0.x or Version 6.1 in the same cell on the same system image (LPAR).

If you have multiple WebSphere Application Server Version 5.0.x nodes, including the deployment manager, on the same LPAR, all nodes must be migrated to WebSphere Application Server Version 6.1 at the same

time. All nodes must be migrated sequentially before starting any WebSphere Application Server Version 6.1 node in the LPAR.

- A WebSphere Application Server Version 6.1 deployment cell can contain mixed releases of Version 5.x or 6.0.x nodes, but there is no mixed-node management support for Version 6.0.1.x.

The Version 6.1 migration tools still migrate these nodes during deployment-manager migration, but they issue a warning message that the nodes cannot be managed by the Version 6.1 deployment manager. You can then do one of the following based on your needs.

- Upgrade all Version 6.0.1.x nodes to at least Version 6.0.2. This will allow them to be administered by a Version 6.1 deployment manager.
- Immediately migrate these nodes to Version 6.1.

Chapter 9. Interoperating

WebSphere Application Server Version 6.1 is interoperable with other WebSphere Application Server versions under certain conditions.

See “Overview of migration, coexistence, and interoperability” on page 77 and “Premigration considerations” on page 81.

WebSphere Application Server Version 6.1 is generally interoperable with WebSphere Application Server Version 5.x and Version 6.0.x. You should apply the latest fix level to support interoperability. Currently, there are no interim fixes necessary to support interoperability. In order to interoperate between WebSphere Application Server for z/OS and a WebSphere Application Server product on a non-z/OS machine, however, you must take the following steps:

1. On the non-z/OS machine, the property `-Dcom.ibm.CORBA.ORBCharEncoding=ASCII` should be set in the administrative console under **server_name > Java and Process Management > Process Definition > Java Virtual Machine > Generic JVM arguments**.
2. On the non-z/OS machine, the property `com.ibm.CORBA.ORBCharDefault = UCS2` should be defined in the administrative console under **server_name > Container Services > ORB Service > Custom Properties**.
3. If you are running WebSphere Application Server for z/OS with security enabled, you must set the property `com.ibm.CORBA.validateBasicAuth=false` in `sas.client.props` on the non-z/OS machine. It is set to true by default.
4. Upgrade the Software Development Kit (SDK) used to one supported by Version 6.1.x.
See Recommended fixes for WebSphere Application Server.

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 10. Configuring ports

When you configure WebSphere Application Server resources or assign port numbers to other applications, you must avoid conflicts with other assigned ports. In addition, you must explicitly enable access to particular port numbers when you configure a firewall.

1. Review the port number settings, especially when you are planning to coexist.
2. **Optional:** Change the port number settings.

You can set port numbers when configuring (customizing) the product after installation. Start thinking about port numbers during the planning phase.

Chapter 11. Troubleshooting migration

You might encounter problems while migrating from an older version of WebSphere Application Server.

If you encounter a problem when you are migrating from a previous version of WebSphere Application Server to Version 6.1, check your log files and other available information.

- Job fails in the Verify step.

This indicates that a configuration error was detected before beginning the migration process. This can be due either to incorrect data entered in the Customization Dialog or a configuration problem. Review the log output for the error detected, then correct and rerun. The logs are located in *temporary_directory_location/nnnnn*, where *temporary_directory_location* is the value that you specified in the Customization Dialog (where the default is */tmp/migrate*) and *nnnnn* is a unique number generated during the creation of your migration jobs and displayed in the Customization Dialog as well as in the JESOUT DDNAME of the WROUT and WRERR steps of your batch job stream.

- Job fails after the Verify step.

In the event of failure in the migration job after the Verify step, you can rerun the migration job, but first you must delete the WebSphere Application Server for z/OS configuration home directory created in the CRHOME step. This corresponds to the home directory you entered in the Customization Dialog, and it can also be found in the migration JCL environment variable *V6_HomeDir*. Because the migration procedure creates a new configuration file system for each node being migrated, it is a simple process to delete the configuration and start from scratch.

- Problems occur with a managed (federated) node migration.

A federated node is the most complex node to migrate because it is essentially two migrations rolled into one. A federated node requires a migration of the node configuration information contained in the deployment manager's master repository as well as the configuration information contained in the federated node. Federated node migration requires an active connection to the deployment manager. If you have security enabled, it is essential that you follow the instructions that were generated from your Customization Dialog session. The migration job must be submitted with a WebSphere Administrator's user ID that has been properly configured for obtaining secure connections.

- Job fails during the application-installation phase of migration.

If you select the option for the migration process to install the enterprise applications that exist in the Version 5.x or Version 6.0.x configuration into the new Version 6.1 configuration, you might encounter error messages during the application-installation phase of migration.

The applications that exist in the Version 5.x or Version 6.0.x configuration might have incorrect deployment information—usually, invalid XML documents that were not validated sufficiently in previous WebSphere Application Server runtimes. The runtime now has an improved application-installation validation process and will fail to install these malformed EAR files. This results in a failure during the application-installation phase of **WASPostUpgrade** and produces an "E:" error message. This is considered a "fatal" migration error.

If migration fails in this way during application installation, you can do one of the following:

- Fix the problems in the Version 5.x or Version 6.0.x applications, and then remigrate.
- Proceed with the migration and ignore these errors.
 1. Restart the migration job in the FINISHUP step to allow the remaining migration functions to be performed.

Do this by adding the **RESTART=FINISHUP** parameter to the job card and resubmitting the job.
 2. Later, fix the problems in the applications and then manually install them in the new Version 6.1 configuration using the administrative console or an install script.

- Out of space errors occur.

The migration logs are located in *temporary_directory_location/nnnnn*, where *temporary_directory_location* is the value that you specified in the Customization Dialog (where the

default is /tmp/migrate) and *nnnn* is a unique number generated during the creation of your migration jobs. Normally, the space requirements for the migration logs are small. However, if you enable tracing, the log files can be quite large. The best practice is to enable tracing only after problems have been found. If tracing is required, try to only enable tracing related to the step in the process that is being debugged. This will help to reduce the space requirements.

Tracing can be enabled using the Customization Dialog, or by changing the variables in the migration JCL from disabled to enabled:

```
TraceState=enabled
profileTrace=disabled
preUpgradeTrace=disabled
postUpgradeTrace=enabled
```

During migration, a backup copy of your Version 5.x or 6.0.x configuration is made. This backup becomes the source of the information being migrated. The default backup location is /tmp/migrate/*nnnn*. This location can be changed in the Customization Dialog. Depending on the size of the node being migrated, this backup can be quite large. If your temporary space is inadequate, then you will need to relocate this backup.

- Batch job time is exceeded.

Each z/OS installation is different with respect to job classes and time limitations. Make sure you have specified appropriate job classes and timeout values on your job card.

- Failures occur during server startup after migration.

Review the generated instructions from the Customization Dialog. Verify that the JCL procedures have been copied over correctly to your PROCLIB, the RACF definitions have been created, the Version 6.1 libraries have been authorized, and, if required, your STEPLIB statements to the Version 6.1 libraries have been specified. Make sure that the daemon process associated with your cell is at the appropriate level. The daemon process must be at the highest WebSphere Application Server for z/OS version level of all servers that it manages within the cell.

After migration, carefully review the job output and log files for errors.

If you migrate a node to Version 6.1 then discover that you need to revert back to Version 5.x or 6.0.x, see “Rolling back your environment” on page 131.

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. For current information available from IBM Support on known problems and their resolution, see the IBM Support page. IBM Support also 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.

Appendix. Directory conventions

References in the product information to *app_server_root*, *profile_root*, and other directories imply specific directory locations. This topic describes the conventions in use for WebSphere Application Server for z/OS.

smpe_root

Refers to the root directory for product code installed with SMP/E.

The corresponding product variable is `smpe.install.root`.

The default is `/usr/lpp/zWebSphere/V6R1`.

configuration_root

Refers to the mount point for the configuration file system (formerly, the configuration HFS) in WebSphere Application Server for z/OS.

The `configuration_root` contains the various `app_server_root` directories and certain symbolic links associated with them. Each different `configuration_root` normally requires its own cataloged procedures under z/OS.

The default is `/WebSphere/V6R1`.

app_server_root

Refers to the top directory for a WebSphere Application Server node.

The node may be of any type—application server, deployment manager, or unmanaged for example. Each node has its own `app_server_root`. Before Version 6.0 of the product information, this was referred to as the "WAS_HOME" directory. Corresponding product variables are `was.install.root` and `WAS_HOME`.

The default varies based on node type. Common defaults are `configuration_root/Appserver` and `configuration_root/DeploymentManager`.

profile_root

Refers to the home directory for a particular instantiated WebSphere Application Server profile.

Corresponding product variables are `server.root` and `user.install.root`.

In general, this is the same as `app_server_root/profiles/profile_name`. On z/OS, this will be always be `app_server_root/profiles/default` because only the profile name "default" is used in WebSphere Application Server for z/OS.

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 Intellectual Property & Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks and service marks

For trademark attribution, visit the IBM Terms of Use Web site (<http://www.ibm.com/legal/us/>).