WebSphere® Application Server for z/OS, Version 6.0.1

IBM

Migrating, coexisting, and interoperating

SA23-2207-00

Note

Before using this information, be sure to read the general information under “Notices” on page 127.

Compilation date: March 16, 2005

Contents

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

What is new for installers

This topic provides an overview of new and changed features in installation, migration, coexistence, and interoperability support. See the *Installing your application serving environment* PDF book for more information.

Presentations from Education on Demand (http://www.ibm.com/developerworks/websphere/library/tutorials/ondemand/)

- Migration overview (ftp://ftp.software.ibm.com/software/eod/WAS_6-0/Install_Migration/Presentations/WASv6_Migration/playershell.swf)

Contents of this section: Migrating, coexisting, and interoperating

Deprecated and removed features

This section summarizes deprecated features in WebSphere Application Server product offerings beginning with Version 6.0 and ending with Version 5.0. As they become available, links to additional information will be provided to help you migrate away from deprecated features.

IBM deprecated and removed features

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

- Deprecation list
- 6.1+ Removal list

Deprecation list

The following tables summarize what is deprecated, by version and release. Each table represents on what version and release the deprecation took effect and what is being deprecated, such as features, APIs, scripting interfaces, tools, wizards, publicly exposed configuration data, naming identifiers, or constants. Where possible, the recommended migration action is provided.

This article consists of the following versions and releases:

- 6.1 + Deprecated features in V6.0.1
- Deprecated features in V6.0
- Deprecated features in V5.1.1
- Deprecated features in V5.1.0.2
- Deprecated features in V5.1
- Deprecated features in V5.0.2
- Deprecated features in V5.0.1
- Deprecated features in V5.0

Deprecated features in V6.0.1

Security features
In future releases, IBM will no longer ship or support the z/OS Secure Authentication Service (z/SAS) IIOP security protocol. It is suggested that you use the Common Secure Interoperability version 2 (CSIv2) protocols. **Recommended migration action:** Use the Common Secure Interoperability version 2 (CSIv2) protocols.

Deprecated features in V6.0

Application programming model and container support features

Support for the following **tsx** tags in the JSP engine is deprecated:

- repeat
- dbconnect
- dbquery
- getProperty
- userid
- passwd
- dbmodify

Recommended migration action:

Instead of using the **tsx** tags, you should use equivalent tags from the JavaServer Pages Standard Tag Library (JSTL). JSTL is supported in WebSphere Application Server V6.0, and the tag library is shipped with the product. Use the following table as a guideline for converting **tsx** tags to JSTL tags:

tsx tag --> ***JSTL tag***

- tsx:repeat --> c:forEach
- tsx:dbconnect --> sql:setDataSource
- tsx:dbquery --> sql:query
- tsx:getProperty --> use standard EL syntax, for example, `c:out value="${book.title}"`, where book is the current index in the result set
- tsx:userid --> use the *user* attribute of the setDataSource tag
- tsx:passwd --> use the *password* attribute of the setDataSource tag
- tsx:dbmodify --> sql:update

Application services features

The WebSphere JRAS Extensions API is deprecated in this release. No further enhancements are planned for JRAS support.

Recommended migration action:

Begin moving over to the java.util.logging package (JSR47) topic (in the *Developing and deploying applications* PDF book), particularly for any new code you are writing.

The UDDI version 2 EJB interface to the UDDI Registry is deprecated.

Recommended migration action:

There is no replacement for the EJB interface. This interface is included in WebSphere Application Server V6.0 for compatibility with V5.x. Users do not need to take any specific actions, and can continue to use the version 2 EJB API, but should be aware that it does not include any UDDI functionality that is new to UDDI version 3, and that the interface may be removed in a future release of WebSphere Application Server.

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

Recommended migration action:

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

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

Recommended migration action:

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

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

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

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

Recommended migration action:

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

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

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

Recommended migration action:

None.

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

- Container-Managed Authentication Alias
- DefaultPrincipleMapping

Recommended migration action:

None.

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

- ChainedRequest
- ChainedResponse
- ChainerServlet
- ServletChain

Recommended migration action:

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

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

Recommended migration action:

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

javax.servlet.*filters* maintain all the functionality of MIME filters. javax.servlet.*filters* are standard APIs, and are supported by all compliant application servers. Refer to the Servlet 2.3 specification or Servlet filtering in the *Administering applications and their environment* PDF book for more information.

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

Recommended migration action:

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

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

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

Recommended migration action:

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

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

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

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

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

Recommended migration action:

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

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).

Recommended migration action:

Instead of using deployment descriptor extensions, you should use the reload enable and interval options provided during application deployment. See Enterprise application settings in the *Administering applications and their environment* PDF book for additional details.

The com.ibm.websphere.servlet.session.UserTransactionWrapper API is deprecated.

Recommended migration action:

There is no replacement for this API. The UserTransaction object can be placed directly into the HTTP session without using a wrapper.

Security features
SOAP-Security (XML digital signature) based on Apache SOAP implementation is deprecated. **Recommended migration action:** Instead of using SOAP-Security, you should migrate your application to JSR-109 implementation of Web service. Also, migrate (reconfigure your application) to use WSS (Web Service Security) 1.0 implementation.

WSS (Web Service Security) draft 13 specification-level support is deprecated in favor of the WSS 1.0 implementation.
 	
Recommended migration action:
 	
Applications should be migrated to the supported WSS 1.0 standard. The draft-level support does not provide interoperability with some third party vendors, as the message level has been changed between the draft and the WSS 1.0 implementation.
 	
WSS 1.0 is only supported in J2EE 1.4 applications. Hence, you need to migrate applications to J2EE 1.4 first. The next step is to use AST/RAD tooling to reconfigure WSS for the migrated application. There is no automatic migration of WSS in this release of AST/RAD tooling for V6.0; the migration has to be done manually.
 	
The following SPI has also been deprecated:
`com.ibm.wsspi.wssecurity.config.KeyLocator`
 	
You need to migrate your implementation to the new SPI for WSS 1.0 support in V6.0:
`com.ibm.wsspi.wssecurity.keyinfo.KeyLocator`
 	
Finally, the JAAS LoginModule implementation needs to be migrated to the new programming model for JAAS LoginModule in V6.0.

In future releases, IBM will no longer ship or support the Secure Authentication Service (SAS) IIOP security protocol. It is suggested that you use the Common Secure Interoperability version 2 (CSIv2) protocols. **Recommended migration action:** Use the Common Secure Interoperability version 2 (CSIv2) protocols.
System administration features
Configuring resources under cell scope is deprecated. **Recommended migration action:** You should configure resources under cluster scope instead. In previous releases, you configured cell scope resources to allow the cluster members to share the resource configuration definition. In Version 6, cell scope resource configuration is discouraged because cell scope resources are visible to every node in the cell, even though not every node in the cell is able to support the resource.
The depl.extension.reg and installdir options for the install command in the AdminApp scripting object are deprecated. **Recommended migration action:** There is no replacement for the depl.extension.reg option. In V5.x, this option was a no-op. For the installdir option, use the installed.ear.destination option instead.
Performance features

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

Recommended migration action:

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

Deprecated features in V5.1.1

Application programming model and container support features
The Web services gateway customization API is deprecated. **Recommended migration action:** No action is required at this time. However, use of Java API for XML-based Remote Procedure Call (JAX-RPC) handlers is recommended over Web services gateway-specific interfaces, such as filters, where possible. The Web services gateway API will be replaced in a future release.
Application services features
The following JDBC drivers are deprecated: • MS SQL Server 2000 Driver for JDBC • SequeLink JDBC driver for MS SQL Server **Recommended migration action:** If you are using either of these JDBC drivers and still want to use MS SQL Server as their database, you can switch to the Connect JDBC driver. You can purchase the Connect JDBC driver from DataDirect Technologies, or you can use the Connect JDBC driver shipped with WebSphere Application Server, which is free for use with WebSphere Application Server.

Deprecated features in V5.1.0.2

Table 1.

Application programming model and container support features
The Web services gateway customization API is deprecated. **Recommended migration action:** No action is required at this time. However, use of Java API for XML-based Remote Procedure Call (JAX-RPC) handlers is recommended over Web services gateway-specific interfaces, such as filters, where possible. The Web services gateway API will be replaced in a future release.
No deprecated features.
Application services features
The following JDBC drivers are deprecated: • MS SQL Server 2000 Driver for JDBC • SequeLink JDBC driver for MS SQL Server **Recommended migration action:** If you are using either of these JDBC drivers and still want to use MS SQL Server as their database, you can switch to the Connect JDBC driver. You can purchase the Connect JDBC driver from DataDirect Technologies, or you can use the Connect JDBC driver shipped with WebSphere Application Server, which is free for use with WebSphere Application Server.

Deprecated features in V5.1

Table 2.

Installation and migration tools
The Application Assembly Tool (AAT) used for developing J2EE applications is being replaced by the Assembly Tool (ATk) component of the Application Server Toolkit (ASTk). **Recommended migration action:** 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 which provides an API for efficient reading, manipulating and writing documentation). The currently packaged version of JDOM in WebSphere Application Server will not be packaged in future releases of WebSphere. **Recommended migration action:** Go to JDOM (www.jdom.org) 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.
In future releases, IBM intends to remove the C++ Object Request Broker (ORB), the C++ library for IDL valuetypes and the WebSphere Application Server C++ security client. IBM will no longer ship or support the Common Object Request Broker Architecture (CORBA) C++ Developer Kit. The CORBA technology is a bridge for migration to a Java 2 Platform Enterprise Edition (J2EE) and WebSphere Application Server environment. In addition to the preceding information, the CORBA C++ client feature will be removed from the Application Clients installation image in future releases. **Recommended migration action:** It is recommended that customers migrate to the Object Request Broker (ORB) service for Java technology that ships with WebSphere Application Server. However, there is no equivalent J2EE functionality for the C++ security client or the C++ Valuetype library. Customers that require such functionality must provide or develop their own. For information on the ORB service for Java technology, see Managing Object Request Brokers in the *Developing and deploying applications* PDF book. 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: • CORBA Interoperability Samples documentation (http://www.ibm.com/developerworks/websphere/library/samples/WASV501/corba.html) • IBM WebSphere Application Servers CORBA Interoperability white paper (http://www.ibm.com/support/docview.wss?uid=swg27004340)
IBM Cloudscape 5.1.x. **Recommended migration action:** No action is required at this time.
Servers and clustering features

Table 2. (continued)

IBM HTTP Server (IHS) 1.3.x. **Recommended migration action:** If you are using IHS 1.3.x with modules: • that are shipped as part of IHS 1.3.x packages, you do not need to take any action to migrate those modules. • supplied by a third party (including other IBM products), you need to obtain IHS/Apache 2 versions of these modules from the third party. • that have been customized or are in-house, you need to port these modules to the new IHS/Apache 2 API.
Application programming model and container support features
Bean Scripting Framework (BSF). JSP execution and debugging functionality is being deprecated in WebSphere Application Server version 5.1. **Recommended migration action:** If using the Javascript, Tcl, and Python languages, the functionality will need to be re-architected. If using BSF scripting in your own custom applications, they will be unaffected. Custom scripts written for the WebSphere Application Server admin console will also be unaffected. This functionality will continue to exist in WebSphere Application Server app server release 5.1, and succeeding releases, until version 6.0. If debugging JSPs you may have to restart the app server during Javascript debugging sessions.

Data access programming interfaces in com.ibm.websphere.rsadapter.
 	
Relational resource adapter interface:
 	
(com.ibm.websphere.rsadapter)
 	
Methods have been deprecated in these types:
`com.ibm.websphere.rsadapter.OracleDataStoreHelper`
`public void doSpecialBLobWork(ResultSet rset,`
`InputStream[] data, String[] blobColumnNames)`
`public String assembleSqlString(String[] blobColumnNames,`
`StringBuffer whereClause, String[] varValues, String tableName)`
 	
Recommended migration action:
 	
These relational resource adapter deprecated methods do not impact the application.
Note: You will not need to implement these deprecated methods in their subclasses if you have the subclass of OracleDataStoreHelper class. Those deprecated methods will not be called by the WebSphere Application Server runtime.

Table 2. (continued)

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

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

- Old method signature

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

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

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

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

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

Recommended migration action:

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

Application services features

Data access binaries -- Common Connector Framework:

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

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

Recommended migration action:

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

For more information on the usage (tools and runtime) of the J2EE Connector Architecture see Data access from J2EE Connector Architecture applications in the Information Center.

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

Recommended migration action:

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

Security features

Table 2. (continued)

Security programming interfaces:
• The API is being deprecated for
`com.ibm.websphere.security.auth.WSPrincipal.getCredential()`
Recommended migration action:
Instead of getting the WSCredential from the principal, you should now use one of the following methods to get the Subject which contains the WSCredential:
– The RunAs Subject is the Subject used for outbound requests.
– The Caller subject is the Subject that represents the authenticated caller for the current request.
– The methods to use to get the runAs and caller subjects are
`com.ibm.websphere.security.auth.WSSubject.getRunAsSubject()`
and
`com.ibm.websphere.security.auth.WSSubject.getCallerSubject()`
respectively.
• The interface is being deprecated in
`com.ibm.websphere.security.auth.WSSecurityContext`
Recommended migration action:
Use JAAS for any authentication related functionality.
• The exception is being deprecated in
`com.ibm.websphere.security.auth.WSSecurityContextException`
Recommended migration action:
Use JAAS for any authentication related functionality.
• The class is being deprecated in
`com.ibm.websphere.security.auth.WSSecurityContextResult`
Recommended migration action:
Use JAAS for any authentication related functionality.

The Integrated Cryptographic Services Facility (ICSF) authentication mechanism is deprecated in V5.1 and will be removed in V6.1. **Recommended migration action:** Use the Lightweight Third Party Authentication (LTPA) instead.
System administration features
The following class is deprecated: `com.ibm.websphere.rsadapter.DB2390DataStoreHelper` **Recommended migration action:** If you currently use the DB2390DataStoreHelper class for the DB2 Legacy CLI-based provider when you are accessing data, you should now use the DB2DataStoreHelper class. If you currently use the DB2390DataStoreHelper class for the DB2 Universal JDBC provider driver when you are accessing data, you should now use the DB2UniversalDataStoreHelper class.

Deprecated features in V5.0.2

Table 3.

Application programming model and container support features

Table 3. (continued)

Apache SOAP channel in Web services gateway. **Recommended migration action:** Gateway services should be deployed to the SOAP HTTP channel instead of the Apache SOAP channel. The Endpoint (URL) of the service will be different for this channel and therefore client programs that are talking to the gateway will need to use the new service Endpoint.
Apache SOAP, WEBSJAVA.SOAP: • soap.jar, • wssoap.jar **Recommended migration action:** See “Migrating Apache SOAP Web services to Web Services for J2EE standards” on page 54 for more information.
Application services features

Data access programming interfaces in com.ibm.websphere.rsadapter.
 	
Relational resource adapter interface:
 	
(com.ibm.websphere.rsadapter)
 	
Methods have been deprecated in these types:
• `com.ibm.websphere.rsadapter.DataStoreHelper`
`public int processSQL(java.lang.String.sqlString, int isolevel,`
`boolean addForUpdate, boolean addextendedForUpdateSyntax);`
`public DataStoreAdatperException mapException(DataStoreAdapterException e);`
• `com.ibm.websphere.rsadapter.GenericDataStoreHelper`
`public int processSQL(java.lang.String.sqlString, int isolevel,`
`boolean addForUpdate, boolean addextendedForUpdateSyntax);`
`public DataStoreAdatperException mapException(DataStoreAdapterException e);`
• `com.ibm.websphere.rsadapter.WSCallHelper`
`public static DataStoreHelper createDataStoreHelper(String dsClassName)`
 	
Recommended migration action:
 	
These relational resource adapter deprecated methods do not impact the application.
Note: You will not need to implement these deprecated methods in their subclasses if you have the subclass of GenericDataStoreHelper. Those deprecated methods will not be called by the WebSphere Application Server runtime.
 	
For com.ibm.websphere.rsadapter.WSCallHelper, please use the getDataStoreHelper(datasource) method to get a DataStoreHelper object.

System administration features
The DB2390DataStoreHelper and the DB2390LocalDataStoreHelper classes. **Recommended migration action:** The DB2DataStoreHelper class now gives all the required helper information needed for the providers that currently use the DB2390DataStoreHelper and the DB2390LocalDataStoreHelper classes.
The DB2 390 Local JDBC Provider (RRS). **Recommended migration action:** This provider is replaced by the DB2 zOS Local JDBC Provider (RRS).

Table 3. (continued)

The **testConnection** command in the AdminControl scripting object (`$AdminControl TestConnection configId props`) is deprecated. Running this command in WebSphere Application Server, Version 5.0.2 or later returns the following message: `WASX7390E: Operation not supported - testConnection command with config id and properties arguments is not supported. Use testConnection command with config id argument only.` **Recommended migration action:** As of WebSphere Application Server, Version 5.0.2 or later, the preferred way to test a data source connection is the testConnection command passing in the data source configuration ID as the only parameter.
The **getPropertiesForDataSource** command in the AdminControl scripting object (`$AdminControl getPropertiesForDataSource configId`) is deprecated. This command incorrectly assumes the availability of the configuration service when you run it in the connected mode. Running this command in WebSphere Application Server, Version 5.0.2 or later returns the following message: `WASX7389E: Operation not supported - getPropertiesForDataSource command is not supported.` **Recommended migration action:** There is no replacement for this command.

Deprecated features in V5.0.1

Table 4.

Application services features
Data access programming interfaces in com.ibm.websphere.rsadapter. Relational resource adapter interface: (com.ibm.websphere.rsadapter) Methods have been deprecated in these types: • `com.ibm.websphere.rsadapter.DataStoreHelper` `public int processSQL(java.lang.String sqlString, int isolevel);` • `com.ibm.websphere.rsadapter.GenericDataStoreHelper` `public int processSQL(java.lang.String sqlString, int isolevel);` • `com.ibm.websphere.rsadapter.DB2390DataStoreHelper` `public int processSQL(java.lang.String sqlString, int isolevel);` **Recommended migration action:** These relational resource adapter deprecated methods do not impact the application. **Note:** You will not need to implement these deprecated methods in their subclasses if you have the subclass of com.ibm.websphere.rsadapter.GenericDataStoreHelper. Those deprecated methods will not be called by the WebSphere Application Server runtime.

Deprecated features in V5.0

Table 5.

Application services features

Table 5. (continued)

The following three methods from com.ibm.websphere.appprofile.accessintent.AccessIntent are deprecated:
`public boolean getPessimisticUpdateHintWeakestLockAtLoad();`
`public boolean getPessimisticUpdateHintNoCollision();`
`public boolean getPessimisticUpdateHintExclusive();`
This is a base api.
Recommended migration action:
Rather than using the three deprecated methods on the AccessIntent interface, developers should use the following method from the same interface:
`public int getPessimisticUpdateLockHint();`
The possible return values are defined on the AccessIntent interface:
`public final static int PESSIMISTIC_UPDATE_LOCK_HINT_NOCOLLISION = 1;`
`public final static int PESSIMISTIC_UPDATE_LOCK_HINT_WEAKEST_LOCK_AT_LOAD = 2;`
`public final static int PESSIMISTIC_UPDATE_LOCK_HINT_NONE = 3;`
`public final static intPESSIMISTIC_UPDATE_LOCK_HINT_EXCLUSIVE = 4;`

Web application programming interfaces -- Various Version 5 methods in com.ibm.websphere.ServletErrorReport
Performance features
Performance Monitoring Infrastructure -- Various Version 5 public methods in: • com.ibm.websphere.pmi.stat.StatsUtil **Recommended migration action:** There is no replacement for StatsUtil. • com.ibm.websphere.pmi.PmiJmxTest **Recommended migration action:** Use PmiClient.findConfig(). • com.ibm.websphere.pmi.client.PmiClient **Recommended migration action:** The getNLSValue (String key) is replaced by getNLSValue (String key, String moduleID).

Removal list

The following tables summarize what is removed by version and release. Use these tables to identify which deprecated items are removed.

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

- 6.1+ Removed features in V6.0.1
- Removed features in V6.0

Removed features in V6.0

Table 6.

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.GlobalIdImpl • 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.GlobalId • 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

Table 6. (continued)

Component	Classes and interfaces
activity	• 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
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

Table 6. (continued)

Component	Classes and interfaces
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	**Method:** • 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

Migrating and coexisting

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

If you have a previous version, you must 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 V6.0.1 installation, the two versions are *coexisting*. The migration articles describe migration from WebSphere Application Server for z/OS V5.x to WebSphere Application Server for z/OS V6.0.1.

You can coexist with or migrate from a previous version of WebSphere Application Server for z/OS.

For information on migrating to V6.0.1, see Chapter 4, “Migrating product configurations,” on page 89. For more information on coexistence among releases, see “Coexistence support” on page 121.

Migration and coexistence overview

WebSphere Application Server provides support for migrating from V5.x to V6.0.1.

New in V6.0.1 is the ″Migrate V5.x Nodes to V6 Nodes″ option off of the main Customization Dialog panel. See Chapter 4, “Migrating product configurations,” on page 89 as a starting point for your case-specific V6.0.1 migration.

While the migration process has been simplified in V6.0.1, there are still restrictions of which you need to be aware:

1. V6.0.1 does not provide support for migrating a V5.x configuration that contains an internal JMS provider in V5.x. The JMS applications must be manually installed and reconfigured post-migration to use the V6.0.1 JMS provider.
2. Certain programming model extensions (PMEs) previously contained in WebSphere Business Integration Server Foundation have been included in the V6.0.1 runtime. Some unsupported PMEs will not be migrated. See “Migrating WebSphere programming model extensions (PMEs)” on page 25 for more information.
3. V5.0.x cannot coexist with V6.0.1 in the same cell on the same system image (LPAR). If you have multiple V5.0.x nodes, including the deployment manager node, on the same LPAR, all nodes MUST be migrated to V6.0.1 at the same time. All nodes must be migrated sequentially before starting any V6.0.1 node in the LPAR.
4. If you are currently running your V5.0.x node using STEPLIB, you must verify that the setupCmdLine.sh contains the STEPLIB of the load libraries. Some 5.0.x nodes do not contain the installation-generated STEPLIB statement. In these instances, you must add the STEPLIB manually to the setupCmdLine.sh prior to running the migration utilities. Here is an example of a setupCMDLine.sh in a properly configured system:

   ```
   STEPLIB=BOSS.VICOM.W000020.SBBOLPA:$STEPLIB
   STEPLIB=BOSS.VICOM.W000020.SBBOLD2A:$STEPLIB
   STEPLIB=BOSS.VICOM.W000020.SBBOLOAD:$STEPLIB
   export STEPLIB
   ```

5. 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 V6.0.1 migration utilities.
6. WebSphere Application Server for z/OS does not support the WASPreUpgrade, WASPostUpgrade, and WASProfile command-line tools. You must use the Customization Dialog to generate the migration jobs.

The remaining migration articles assume that V6.0.1 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 V6.0.1. Prior levels of WebSphere Application Server for z/OS will continue to run at the higher prerequisite levels. For more information, see “Prerequisites needed for WebSphere Application Server for z/OS” on page 23.
- Set up V6.0.1 to eliminate potential LPA conflicts with a prior V5.x install. V6.0.1 and V5.x both require the placement of some code into LPA (SBBOLPA). In addition, it is recommended that additional product code (SBBOLOAD) be placed into LPA for performance reasons. Because of naming conflicts, V6.0.1 and V5.x product code can not be in LPA at the same time. To support coexistence, be sure you properly set up LPA and STEPLIB.
- Utilize WLM DAE when configuring V6.0.1 to allow both the use of a specific server name by V6.0.1 and by a server on V5.x.
- Review the ports that have been defined to ensure that the V6.0.1 installation does not conflict. In particular, when installing to coexist with V5.x, note that the default daemon port definition for both V6.0.1 and V5.x is the same. See z/OS port assignments in the *Installing your application serving environment* PDF book for default port information.

WebSphere Application Server for z/OS V6.0.1 continues to support HTTP transports, while adding support for transport chains. For more information on WebSphere Application Server for z/OS V6.0.1 transport support, see the following topics in the *Administering applications and their environment* PDF book:

- Transports
- Configuring HTTP transports
- HTTP transport collection
- HTTP transport settings
- HTTP transport custom properties

When you are ready to begin the migration process, see Chapter 4, “Migrating product configurations,” on page 89, which is a starting point for the planning information, Customization Dialog walkthroughs, and V5.x to V6.0.1 migration explanations for base Application Server nodes, deployment managers, and federated nodes.

Configuration mapping during migration

This topic describes what changes during migration, which is 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 Version 6 migration tools maps objects and attributes to the Version 6 environment when you restore a configuration from a previous version.

Bootstrap port

Migration maps a default bootstrap NameServer port setting, 900, from V5.x to the V6 NameServer default, 2809. The migration tools map a non-default value directly into the V6 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, such as memory heap sizes, are not migrated because their roles in the V6 configuration either do not exist, have different meanings, or have different scopes.

GenericServer

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

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

Migration of a V5.x node to a V6 node

You can migrate a V5.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 V5.x to V6. If you use a different cell name, federated nodes cannot successfully migrate to the Network Deployment V6 cell.

Migrating a base WebSphere Application Server node that is within a cell to V6 also migrates the node agent to V6. A cell can have some V6 nodes and other nodes that are at V5.x levels.

Name bindings

Version 6 has a new naming structure. All references, such as Enterprise JavaBeans (EJB) references that were valid in previous versions no longer work in Version 6. However, you can use the administrative console to add a name binding that maps an old name into the new Version 6 naming structure. For example, the name of the Version 5.0.x enterprise bean reference can be both the name of the binding and the Java Naming and Directory Interface (JNDI) name in the Version 6 name space.

Note: The contents of name spaces from previous versions are not automatically migrated to Version 6.

Policy file migration from Version 5.x to Version 6

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

- Any comments located in the V6 policy file will be lost. They are replaced with the comments contained in the V5 policy file.
- Migration will NOT attempt to merge permissions or grants; it is strictly an add-type migration. If the permission or grant is not located in the V6 file, the migration will bring it over.
- Security is a critical component; thus, the migration makes any additions at the end of the original .policy file right after the comment `MIGR0372I: Migrated grant permissions follow`. This is done to help administrators verify any policy file changes that the migration has made.

Properties directory and the lib/app directory

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

Property file migration from Version 5.x to Version 6

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

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

Migration does not overlay property files.

Resource Adapter Archive (RAR) referenced by J2C resources

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

Samples

During the migration of the deployment manager, V5.x Samples for federated nodes are migrated. Equivalent Version 6 Samples are available to use for all other V5.x Samples.

Servlet package name changes

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

Stdin, stdout, stderr, passivation, and working directories

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

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

Transport ports

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

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

- **-scriptCompatibility**="true": This results in your transport ports being brought over as they are (this is the default).
- **-scriptCompatibility**="false": This results in the transports ports being converted to the new implementation of channels and chains. From an external application usage standpoint, they will still act the same, but they have been moved to the TransportChannelService. For further information on transport chains and channels, see Transport chains in the *Administering applications and their environment* PDF book.

Web modules

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

Version 5.x to Version 6 migration

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

Java heap size for migrating EAR files

When migrating all V5.x EAR files to V6 using the wsadmin tool, the WASPostUpgrade tool uses the default maximum Java heap size value of 64MB to install the EAR files.

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

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

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

Installing the application on WebSphere Application Server, Version 6

Assume that:

Installation root
: `C:\WebSphere\AppServer`

Number signs (###)
: Maximum heap size value

EAR_file_name
: The name of the EAR file

app_name
: The name of the application.

server_name
: The name of the server on which the EAR file installs

node_name
: The name of the node on which the server is configured

The command appears on more than one line for clarity.

```
wsadmin -conntype NONE
        -javaoption
        -Xmx###m
        -c "$AdminApp install
               C:\\WebSphere\\AppServer\\installableApps\\
```

```
          EAR_file_name
        {-nodeployejb
         -appname app_name
         -server server_name
         -node node_name}"
```

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

Assume that:

Installation root
: `C:\WebSphere\DeploymentManager`

Number signs (###)
: Maximum heap size value

EAR_file_name
: The name of the EAR file

app_name
: The name of the application.

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

The command appears on more than one line for clarity.

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

Specification level summary of WebSphere Application Server

This article shows the various releases of the WebSphere Application Server for z/OS family and the specification and functional differences between them.

Table 7. Specification and functional differences between WebSphere Application Server for z/OS releases

Specifications	V5	V5.1	V6.0.1

Table 7. Specification and functional differences between WebSphere Application Server for z/OS releases (continued)

Java related	• Requires SDK 1.3 • Supports J2EE 1.3 levels: – Servlet 2.3 – JSP 1.2 – JDBC 2.0 – EJB 2.0 – part of J2SE 1.3 – JTA 1.0 – part of J2SE 1.3 – JMS 1.0.2 – JavaMail 1.2 – JAP 1.0 – Client container – Java Authentication and Authorization Service (JAAS) 1.0 – Java API for XML Parsing (JAXP) 1.1 – J2EE Connector Architecture 1.0	• Requires SDK 1.4.1 • Supports J2EE 1.4.1 levels: – Servlet 2.3 – JSP 1.2 – JDBC 2.0 – EJB 2.0 – part of J2SE 1.3 – JTA 1.0 – part of J2SE 1.3 – JMS 1.0.2 – JavaMail 1.2 – JAP 1.0 – Client container – Java Authentication and Authorization Service (JAAS) 1.0 – Java API for XML Parsing (JAXP) 1.1 – J2EE Connector Architecture 1.0	• Requires SDK 1.4.2 • Supports J2EE 1.4.1 levels: – Servlet 2.3 – JSP 1.2 – JDBC 2.0 – EJB 2.0 – part of J2SE 1.3 – JTA 1.0 – part of J2SE 1.3 – JMS 1.0.2 – JavaMail 1.2 – JAP 1.0 – Client container – Java Authentication and Authorization Service (JAAS) 1.0 – Java API for XML Parsing (JAXP) 1.1 – J2EE Connector Architecture 1.0
Web Services	• UDDI V2 • SOAP 2.3	• UDDI V2 • SOAP 2.3	• UDDI V2 • SOAP 2.3

- **WebSphere Application Server for z/OS V5.0.x** provides support for next generation technologies-J2EE 1.3 compatible with support for key Web services. V5.0.x is Network Deployment compliant and continues to build towards WebSphere Application Server for z/OS family consistency in architecture, administrave console, and programming APIs.
- **WebSphere Application Server for z/OS V5.1 and V6.0.1** continue to support WebSphere Application Server for z/OS brand consistency. Additionally, these products provide complete client and server support for J2EE 1.4.1.

Prerequisites needed for WebSphere Application Server for z/OS

The following table describes the prerequisites needed for WebSphere Application Server for z/OS V5, V5.1, and V6.0.1.

Table 8. Prerequisites needed for WebSphere Application Server for z/OS V5.0.x, V5.1, and V6.0.1.

Pre-Reqs	V5.0.x	V5.1	V6.0.1
SDK (included with WebSphere Application Server for z/OS)	SDK 1.3	SDK 1.4.1	SDK 1.4.2
OS/390 or z/OS	R10 and above	z/OS 1.2 or higher	z/OS 1.4 or higher
	z/OS 1.2 is required for the Dynamic application environment	z/OS.e 1.3 or higher	z/OS.e 1.4 or higher

Table 8. Prerequisites needed for WebSphere Application Server for z/OS V5.0.x, V5.1, and V6.0.1. (continued)

HTTP Server	Must use HTTP transport Options for connecting to HTTP transport: • Direct Browser • IBM HTTP Server for OS/390 (R10) • Any HTTP Server supported by V5.0 family plugins	Must use HTTP transport Options for connecting to HTTP transport: • Direct Browser • IBM HTTP Server for z/OS 1.2 • Any HTTP Server supported by V5.1 family plugins	Must use HTTP transport Options for connecting to HTTP transport: • Direct Browser • IBM HTTP Server for z/OS 1.4 • Any HTTP Server supported by V6.0.1 family plugins
Sysplex	Required	Required	Required
OS/390 Communications server (TCP/IP)	Required	Required	Required
OS/390 Unix System Services and Hierarchical file system (HFS)	Required Shared HFS is supported, but is no longer required for a multiple image sysplex	Required Shared HFS is supported, but is no longer required for a multiple image sysplex	Required Shared HFS is supported, but is no longer required for a multiple image sysplex
SecureWay Security Server (RACF) or Equivalent	Required	Required	Required
System Logger	Required	Required	Required
LightWeight Directory Access Protocol (LDAP) Server	Not required	Not required	Not required
Workload Manager in Goal mode	Required Dynamic AE (programmatic setup of WLM application environment) requires PTF against z/OS 1.2	Required Dynamic AE (programmatic setup of WLM application environment) requires PTF against z/OS 1.2	Required
Resource Recovery Services (RRS)	Required	Required	Required
FTP server	Not required	Not required	Not required
System SSL Security	Required for SSL	Required for SSL	Required for SSL
DB2	Not required or supported for SM function V7 required for user data (JDBC compliant driver)	Not required or supported for SM function V7 or higher required for user data (JDBC compliant driver)	Not required or supported for SM function

Table 8. Prerequisites needed for WebSphere Application Server for z/OS V5.0.x, V5.1, and V6.0.1. (continued)

WSMQ	Not required - Integrated JMS provided as part of WAS 5.0 If integrated provider is not required: • Can continue to use the same product stack as in WAS 4.0.1 **Note:** Functions such as XA support are only available with MQ5.3.1. • Can purchase and install full MQ5.3.1 product.	Not required - Integrated JMS provided as part of WAS 5.1 If integrated provider is not required: • Can continue to use the same product stack as in WAS 4.0.1 **Note:** Functions such as XA support are only available with MQ5.3.1. • Can purchase and install full MQ5.3.1 product.	Not required - Integrated JMS provided as part of WAS 6.0.1 If integrated provider is not required: • Can purchase and install full MQ5.3.1 product.
IMS JCA	IMS Connect 2.1(requires IMS V8.1)	IMS Connect 2.1(requires IMS V8.1)	IMS Connect 2.2 (requires IMS V8.1) or IMS V9 Integrated Connect function
CICS JCA	CICS Transaction Gateway 5.0.1 (requires CICS TS 1.3)	CICS Transaction Gateway 5.1 (requires CICS TS 1.3)	CICS Transaction Gateway 6.0

Migrating WebSphere programming model extensions (PMEs)

This topic describes the movement of a subset of PMEs to WebSphere Application Server for z/OS Version 6.0.1 from WebSphere Business Integration Server Foundation V5.1.x.

Overview of PME migration

The migration of PME services from WebSphere Business Integration Server Foundation V5.1.x to WebSphere Application Server for z/OS Version 6.0.1 is handled on an individual basis. For PME services that are not supported in WebSphere Application Server for z/OS V6.0.1, all configuration information is removed. For PME services that are supported in WebSphere Application Server for z/OS V6.0.1, 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 for z/OS V6.0.1. Any application that uses a resource for a service that is not supported by WebSphere Application Server for z/OS V6.0.1 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 V6 deployment manager managing WebSphere Business Integration Server Foundation V5.1.x nodes), the first syncNode performed by the system downloads a configuration from the WebSphere Application Server V6 deployment manager to the WebSphere Business Integration Server Foundation V5.1.x nodes. Therefore, any PME service that is not supported by WebSphere Application Server V6.0.1 will be rendered inoperable on the WebSphere Business Integration Server Foundation V5.1.x node.

PME-specific information

For more information on the specific PMEs that have been migrated to WebSphere Application Server for z/OS V6.0.1, see “Learn about WebSphere programming extensions” on page 81.

Mapping of V4.0.1 environment variables to V6.0.1 console settings

The following table lists only V6.0.1 WebSphere Application Server for z/OS variables related to diagnosis, along with their equivalent V4.0.1 environment variables. This information is provided only as an aid to IBM service personnel. The migration utilities do not support migrating from V4.0.1 to V6.0.1.

Warning: Do not use this information to manually modify the contents of a *was.env* file. The *was.env* file is managed by WebSphere Application Server for z/OS, and its content is rewritten with each change made to the WebSphere Application Server for z/OS configuration. Therefore, any hand-editing will be overwritten. To determine which WebSphere Application Server for z/OS variable, custom property or administrative console field must be updated in order to change the value of a specific internal variable, see Changing the values of variables referenced in BBOM0001I messages in the *Administering applications and their environment* PDF book.

Table 9. V4.0.1 environment variables and their equivalent V6.0.1 WebSphere Application Server for z/OS variables

V4.0.1 environment variables	Equivalent V6.0.1 WebSphere Application Server for z/OS variables
BBOC_HTTPALL_TCLASS_FILE	**http_transport_class_mapping_file**
BBOC_HTTP_INPUT_IDENTITY	**protocol_http_defaultIdentity**
BBOC_HTTP_INPUT_TIMEOUT	**protocol_http_timeout_input**
BBOC_HTTP_LISTEN_IP_ADDRESS	**protocol_http_listenIPAddress**
BBOC_HTTP_MAX_PERSIST_REQUESTS	**protocol_http_max_persist_requests**
BBOC_HTTP_OUTPUT_TIMEOUT	**protocol_http_timeout_output**
BBOC_HTTP_OUTPUT_TIMEOUT_RECOVERY	**protocol_http_timeout_output_recovery**
BBOC_HTTP_PERSISTENT_SESSION_TIMEOUT	**protocol_http_timeout_persistentSession**
BBOC_HTTP_PORT	**protocol_http_port**
BBOC_HTTP_SSL_IDENTITY	**protocol_https_default_identity**
BBOC_HTTP_SSL_INPUT_TIMEOUT	**protocol_https_timeout_input**
BBOC_HTTP_SSL_LISTEN_IP_ADDRESS	**protocol_https_listenIPAddress**
BBOC_HTTP_SSL_MAX_PERSIST_REQUESTS	**protocol_https_max_persist_requests**
BBOC_HTTP_SSL_OUTPUT_TIMEOUT	**protocol_https_timeout_output**
BBOC_HTTP_SSL_OUTPUT_TIMEOUT_RECOVERY	**protocol_https_timeout_output_recovery**
BBOC_HTTP_SSL_PERSISTENT_SESSION_TIMEOUT	**protocol_https_timeout_persistentSession**
BBOC_HTTP_SSL_PORT	**protocol_https_port**
BBOC_HTTP_SSL_TRANSACTION_CLASS	**protocol_https_transactionClass**
BBOC_HTTP_TRANSACTION_CLASS	**protocol_http_transactionClass**
BBOC_LOG_RESPONSE_FAILURE	**protocol_bboc_log_response_failure**
BBOC_LOG_RETURN_EXCEPTION	**protocol_bboc_log_return_exception**
BBODUMP	**ras_dumpoptions_dumptype**
BBODUMP_CEE3DMP_OPTIONS	**ras_dumpoptions_ledumpoptons**
BBOLANG	**nls_language**
BBOO_WORKLOAD_PROFILE	**server_region_workload_profile**
CBCONFIG	**config_root**
CLASSPATH	**server_region_classpath**
CLIENT_RESOLVE_IPNAME	Replaced by the corbaloc function in J2EE 1.3 CosNaming INS

Table 9. V4.0.1 environment variables and their equivalent V6.0.1 WebSphere Application Server for z/OS variables (continued)

V4.0.1 environment variables	Equivalent V6.0.1 WebSphere Application Server for z/OS variables
CLIENT_RESOLVE_PORT	Replaced by the corbaloc function in J2EE 1.3 CosNaming INS
CLIENT_TIMEOUT	**protocol_iiop_local_timeout**
CLIENTLOGSTREAM	**client_ras_logstreamname**
CONFIGURED_SYSTEM	**server_configured_system_name**
CTL_LIBPATH	**control_region_libpath**
DAEMON_IPNAME	**protocol_iiop_daemon_listenIPAddress**
DAEMON_PORT	**protocol_iiop_daemon_port**
DAEMON_SSL_PORT	**protocol_iiop_daemon_port_ssl**
DM_GENERIC_SERVER_NAME	**daemonName**
DM_SPECIFIC_SERVER_NAME	**daemonInstanceName**
ENABLE_TRUSTED_APPLICATIONS	**control_region_security_enable_trusted_applications**
GENERIC_SERVER_NAME	**server_generic_short_name**
GENERIC_UUID	**server_generic_uuid**
IIOP_SERVER_SESSION_KEEPALIVE	**protocol_iiop_server_session_keepalive**
IVB_DEBUG_ENABLED	**ras_debugEnabled**
JVM_ENABLE_VERBOSE_GC	No longer a variable. Specify as Java option: -verbose:gc
JVM_HEAPSIZE	No longer a variable. Specify as Java option: -Xmx<size>
JVM_MINHEAPSIZE	No longer a variable. Specify as Java option: -Xms<size>
JVM_LOCALREFS	**control_region_jvm_localrefs**
JVM_LOCALREFS	**server_region_jvm_localrefs**
JVM_LOGFILE	**control_region_jvm_logfileserver_region_jvm_logfile**
LIBPATH	**server_region_libpath**
LOGSTREAMNAME	**ras_log_logstreamName**
MAX_SRS	**wlm_maximumSRCount**
MIN_SRS	**wlm_minimumSRCount**
OTS_DEFAULT_TIMEOUT	**transaction_defaultTimeout**
OTS_MAXIMUM_TIMEOUT	**transaction_maximumTimeout**
RAS_MINORCODEDEFAULT	**ras_minorcode_action**
RECOVERY_TIMEOUT	**transaction_recoveryTimeout**
REM_PASSWORD	**client_protocol_password**
REM_USERID	**client_protocol_user**
SPECIFIC_SERVER_NAME	**server_specific_short_name**
SPECIFIC_UUID	**server_specific_uuid**
SRVIPADDR	**protocol_iiop_listenIPAddress**
SSLIIOP_SERVER_SESSION_KEEPALIVE	**protocol_iiop_server_session_keepalive_ssl**
SSL_KEYRING	**security_sslKeyring**
TRACE_EXCLUDE_SPECIFIC	**ras_trace_exclude_specific**
TRACEALL	**ras_trace_defaultTracingLevel**

Table 9. V4.0.1 environment variables and their equivalent V6.0.1 WebSphere Application Server for z/OS variables (continued)

V4.0.1 environment variables	Equivalent V6.0.1 WebSphere Application Server for z/OS variables
TRACEBASIC	**ras_trace_basic**
TRACEBUFFCOUNT	**ras_trace_BufferCount**
TRACEBUFFLOC	**ras_trace_outputLocation**
TRACEBUFFSIZE	**ras_trace_BufferSize**
TRACEDETAIL	**ras_trace_detail**
TRACEMINORCODE	**ras_trace_minorCodeTraceBacks**
TRACEPARM	**ras_trace_ctraceParms**
TRACESPECIFIC	**ras_trace_specific**
WAS_JAVA_OPTIONS	No longer a variable. Java options are specified in the JVM configuration of the servant process.

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

Migrate product and Web server configurations

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

Determine what configuration information needs to change

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

Documentation

Review the software and hardware prerequisites

Learn which product configurations are supported.

Documentation
(www.ibm.com/software/webservers/appserv/doc/latest/prereq.html)

Migrate your applications

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

Identify deprecated features that you might be using

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

Documentation

Review the software and hardware prerequisites

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

Documentation:
Supported hardware
and software
(www.ibm.com/software/webservers/appserv/doc/latest/prereq.html)

Learn about WebSphere applications

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

Documentation:
Chapter 3, “Learn about WebSphere applications,” on page 33

Coexist

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

Review supported coexistence scenarios

All combinations of V4.x products, V5.x products, and V6.0 products can coexist so long as there are no port conflicts. There are some coexistence limitations for V5 products that have the embedded messaging feature installed, as described in the documentation.

Documentation:
Chapter 5, “Coexisting,” on page 121

Obtain valid port settings for coexistence

Set port numbers while you are customizing the product after installation.

Documentation:

- Planning a TCP/IP port convention in the *Installing your application serving environment* PDF book

---

Interoperate

Interoperability is exchanging data between two coexisting product installations.

Interoperate across product versions

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

Documentation:
Chapter 6,
“Interoperating,” on
page 123

Chapter 3. Learn about WebSphere applications

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

Web applications

Migrating V6.0 servers from multi-broker replication domains to data replication domains

Use this task to migrate multi-broker replication domains to data replication domains. Multi-broker domains 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 V6.0 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.

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, create an empty data replication domain. Click **Environment > Replication domains > New** in the administrative console.
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.

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 V6.0 application servers use the two types of replication domains:

	V5.x application servers using replication domains	**V6.0 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.
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.

	V5.x application servers using replication domains	V6.0 application servers using replication domains
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.
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.

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

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

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

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

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.

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

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

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

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

   The Version 4.x single path limitation does not exist in Version 5. However, users who choose to use the `com.ibm.servlet.engine.webapp.SimpleFileServlet` in Version 5 must do one of the following:
   - Rename `com.ibm.servlet.engine.webapp.SimpleFileServlet` to `com.ibm.ws.webcontainer.servlet.SimpleFileServlet`.
   - Opena Web deployment descriptor editor using an assembly tool and select **File serving enabled** on the **Extensions** tab.

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

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

2. Use WebSphere Application Server Version 5 package names for any WebSphere Application Server Version 4.0.1 internal servlets, which are implemented in your application.

   In WebSphere Application Server Version 4.0.1, Web modules with a context root setting of / are not supported. Accessing Web modules with this root context results in `HTTP 404 - File not Found` errrors.

   To bypass the errors, and to enable the serving of static files from the root context, WebSphere Application Server Version 4.0.1 users are advised to add the servlet class, `com.ibm.servlet.engine.webapp.SimpleFileServlet`, to their Web module.

   The Version 4.0.1 single path limitation does not exist in Version 5. However, users who choose to use the `com.ibm.servlet.engine.webapp.SimpleFileServlet` in Version 5 must do one of the following:
   - Rename `com.ibm.servlet.engine.webapp.SimpleFileServlet` to `com.ibm.ws.webcontainer.servlet.SimpleFileServlet`.
   - Opena Web deployment descriptor editor in the Assembly Toolkit and select **File serving enabled** on the **Extensions** tab.

   The following list identifies the other internal servlets affected by the Version 5 package name change:
   - DefaultErrorReporter
   - AutoInvoker

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

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

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

4. 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.

   The XML servlet configuration file is not supported in WebSphere Application Server Version 5. The direct use of the servlet has been deprecated. The PageList servlet function is still available but is configured as part of the servlet extension configuration in the WAR file.

5. Set a content type if your servlet generates Hyper Text Markup Language (HTML) output.

   The default behavior of the Web container changed in WebSphere Application Server Version 5. If the servlet developer does not specify a content type in the servlet then the container is forbidden to set one automatically. Without an explicit content type setting, the content type is set to null. The Netscape browser displays HTML source as plain text with a null content type setting.

   To resolve this problem, do one of the following:
   - Explicitly set a content type in your servlet.
   - Opena Web deployment descriptor editor in an assembly tool and select **Automatic Response Encoding enabled** on the **Extensions** tab.

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

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

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

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

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

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

   If you install Version 4.x EAR files, deployed with the `JSP Precompile` option, in Version 5, and you choose not to follow the migration path, do one of the following:
   - Select the `Pre-compile JSP` option in the administrative console Install New Application window.
   - Specify the `-preCompileJSPs` option when using the Wsadmin tool.

8. Migrate WebSphere Version 4.0.1 enterprise applications to Version 5.

   Note: The WebSphere Application Server Version 4.0.1 JSP page implementation class files are not compatible with the WebSphere Application Server Version 5 JSP container.

   You must do one of the following:
   - Select the `Pre-compile JSP` option in the administrative console `Install New Application` window.
   - Specify the `-preCompileJSPs` option when using the Wsadmin tool.

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

   Section 8.2 of the JSP 1.2 specification states:

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

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

   ```
   <%@page import="myBeanClass" %>
                  . . .
   <jsp:useBean id="myBean" class="myBeanClass" scope="session"/>
   ```

   In WebSphere Application Server Version 5, the JSP engine creates JSP page implementation classes in the `org.apache.jsp` package. If a class in the unnamed package is not explicitly imported, then the

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

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

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

        import SimpleName;

is no longer legal. Nor is the syntax

        import ClassInUnnamedNamespace.Nested;

which would import a nested class from the unnamed namespace.
    To fix such problems in your code, move all of the classes from the
unnamed namespace into a named namespace.
```

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

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

Supported open specification levels in WebSphere Application Server Version 6 are documented in article, Migrating.

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

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

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

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. You need to recompile all JSP pages when migrating from WebSphere Application Server Version 5.x to version 6.

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

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

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

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

   The Version 4.x single path limitation does not exist in Version 5. However, users who choose to use the `com.ibm.servlet.engine.webapp.SimpleFileServlet` in Version 5 must do one of the following:
   - Rename `com.ibm.servlet.engine.webapp.SimpleFileServlet` to `com.ibm.ws.webcontainer.servlet.SimpleFileServlet`.
   - Opena Web deployment descriptor editor using an assembly tool and select **File serving enabled** on the **Extensions** tab.
2. Use WebSphere Application Server Version 5 package names for any WebSphere Application Server Version 4.0.1 internal servlets, which are implemented in your application.

   In WebSphere Application Server Version 4.0.1, Web modules with a context root setting of / are not supported. Accessing Web modules with this root context results in `HTTP 404 - File not Found` errors.

   To bypass the errors, and to enable the serving of static files from the root context, WebSphere Application Server Version 4.0.1 users are advised to add the servlet class, `com.ibm.servlet.engine.webapp.SimpleFileServlet`, to their Web module.

   The Version 4.0.1 single path limitation does not exist in Version 5. However, users who choose to use the `com.ibm.servlet.engine.webapp.SimpleFileServlet` in Version 5 must do one of the following:
   - Rename `com.ibm.servlet.engine.webapp.SimpleFileServlet` to `com.ibm.ws.webcontainer.servlet.SimpleFileServlet`.
   - Opena Web deployment descriptor editor in 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.

   The XML servlet configuration file is not supported in WebSphere Application Server Version 5. The direct use of the servlet has been deprecated. The PageList servlet function is still available but is configured as part of the servlet extension configuration in the WAR file.
4. Migrate WebSphere Version 4.0.1 enterprise applications to Version 5.

   Note: The WebSphere Application Server Version 4.0.1 JSP page implementation class files are not compatible with the WebSphere Application Server Version 5 JSP container.

   You must do one of the following:
   - Select the `Pre-compile JSP` option in the administrative console `Install New Application` window.
   - Specify the `-preCompileJSPs` option when using the Wsadmin tool.

Migrating HTTP sessions

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

Migrating from Version 5.x

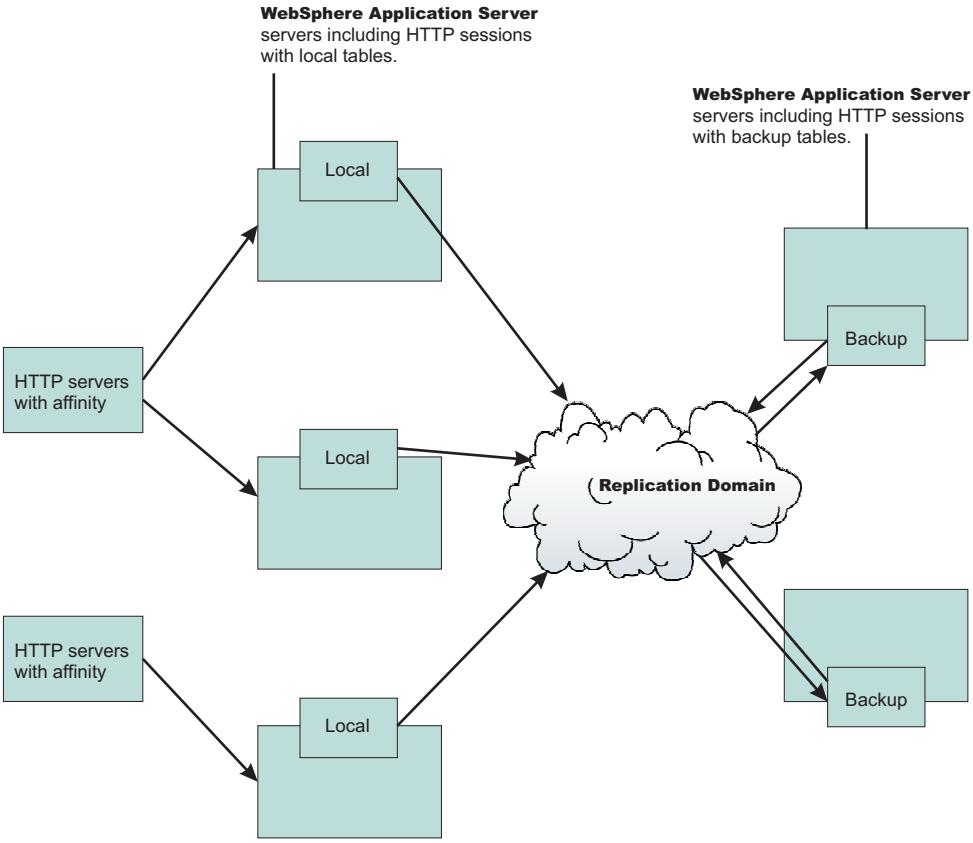
No programmatic changes are required to migrate from version 5.x to version 6.

Migrating from Version 4.0

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

Memory-to-memory topology: Client/server function

The following figure depicts the client/server mode. There is a tier of applications servers that host Web applications using HTTP sessions, and these sessions are replicated out as they are created and updated. There is a second tier of servers without a Web application installed, where the session manager receives updates from the replication clients.



Benefits of the client/server configuration include:

Isolation (for failure recovery)

In this case we are isolating the handling of backup data from local data; aside from isolating the moving parts in case of a catastrophic failure in one of them, you again free up memory and processing in the servers processing the Web application

Isolation for stopping and starting

You can recycle a backup server without affecting the servers running the application (when there are two or more backups, failure recovery is possible), and conversely recycle an application JVM without potentially losing that backup data for someone.

Consolidation

There is most likely no need to have a one-to-one correspondence between servers handling backups and those processing the applications; hence, you are again reducing the number of places to which you transfer the data.

Disparate hardware:

While you run your Web applications on cheaper hardware, you may have one or two more powerful computers in the back end of your enterprise that have the capacity to run a couple of session managers in replication server mode; allowing you to free up your cheaper Web application hardware to process the Web application.

Timing consideration: Start the backup application servers first to avoid unexpected timing windows. The clients attempt to replicate information and HTTP sessions to the backup servers as soon as they come up. As a result, HTTP sessions that are created prior to the time at which the servers come up might not replicate successfully.

Memory-to-memory session partitioning

Session partitioning gives the administrator the ability to filter or reduce the number of destinations that the session object gets sent to by the replication service. You can also configure session partitioning by specifying the number of replicas on the replication domain. The Single replica option is chosen by default. Since the number of replicas is global for the entire replication domain, all the session managers connected to the replication domain use the same setting.

Single replica
You can replicate a session to only one other server, creating a single replica. When this option is chosen, a session manager picks another session manager that is connected to the same replication domain to replicate the HTTP session to during session creation. All updates to the session are only replicated to that single server. This option is set at the replication domain level. When this option is set, every session manager connected to this replication domain creates a single backup copy of HTTP session state information on a backup server.

Full group replica
Each object is replicated to every application server that is configured as a consumer of the replication domain. However, this topology is the most redundant because everyone replicates to everyone and as you add servers, more overhead (both CPU and memory) is needed to deal with replication. This mode is most useful for dynamic caching replication.

Specific number of replicas
You can specify a specific number of replicas for any entry that is created in the replication domain. The number of replicas is the number of application servers that the user wants to use to replicate in the domain. This option eliminates redundancy that occurs in a full group replica and also provides additional backup than a single replica. The number of replicas cannot exceed the total number of application servers in the cluster.

EJB applications

Migrating V6.0 servers from multi-broker replication domains to data replication domains

Use this task to migrate multi-broker replication domains to data replication domains. Multi-broker domains 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 V6.0 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.

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, create an empty data replication domain. Click **Environment > Replication domains > New** in the administrative console.
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.

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 V6.0 application servers use the two types of replication domains:

	V5.x application servers using replication domains	**V6.0 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	V6.0 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	V6.0 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.

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 findBy*Method* 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 findBy*Method* 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 `set`*`Field`*`(`*`value`*`)`.
3. In each CMP bean, create abstract get and set methods for the primary key.
4. In beans with CMP version 1.x, create an EJB Query Language statement for each finder method.

   Note: EJB Query Language has the following limitations in Application Developer Version 5:
   - EJB Query Language queries involving beans with keys made up of relationships to other beans appear as invalid and cause errors at deployment time.

- The IBM EJB Query Language support extends the EJB 2.1 spec in various ways, including relaxing some restrictions, adding support for more DB2 functions, and so on. If portability across various vendor databases or EJB deployment tools is a concern, then care should be taken to write all EJB Query Language queries strictly according to instructions described in Chapter 11 of the EJB 2.1 specification.

5. In finder methods for beans with CMP version 1.x, return java.util.Collection instead of java.util.Enumeration.
6. Update handling of non-application exceptions.
   - To report non-application exceptions, throw javax.ejb.EJBException instead of java.rmi.RemoteException.
   - Modify rollback behavior as needed: In EJB versions 1.1 and 2.1, all non-application exceptions thrown by the bean instance result in the rollback of the transaction in which the instance is running; the instance is discarded. In EJB 1.0, the container does not roll back the transaction or discard the instance if it throws java.rmi.RemoteException.
7. Update rollback behavior as the result of application exceptions.
   - In EJB versions 1.1 and 2.1, an application exception does not cause the EJB container to automatically roll back a transaction.
   - In EJB Version 1.1, the container performs the rollback only if the instance has called setRollbackOnly() on its EJBContext object.
   - In EJB Version 1.0, the container is required to roll back a transaction when an application exception is passed through a transaction boundary started by the container.
8. Update any CMP setting of application-specific default values to be inside ejbCreate (not using global variables, since EJB 1.1 containers set all fields to generic default values before calling ejbCreate, which overwrites any previous application-specific defaults). This approach also works for EJB 1.0 CMPs.

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

Adjusting exception handling for EJB wrappered 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 wrappered in WebSphere Application Server version 5.1 cannot be wrappered 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.

The product uses interoperable versions of some class types to enable interoperability. However, older 4.0.x client and application server versions do not support the interoperability classes, which makes them uninteroperable with versions that use the classes. The system property *com.ibm.websphere.container.portable* remedies this situation by enabling newer versions of the application server to turn off the interoperability classes. This lets a more recent application server return class types that are interoperable with an older client.

Depending on the value of com.ibm.websphere.container.portable, application servers at versions 5 and later, and 4.0.3 and later, return different classes for the following:
- Enumerations and collections returned by EJB 1.1 finder methods
- EJBMetaData
- Handles to:
  - Entity beans
  - Session beans
  - Home interfaces

If the property is set to `false`, application servers return the old class types, to enable interoperability with 4.0.2 and earlier. If the property is set to `true`, application servers return the new classes.

The following tables show interoperability characteristics for various version combinations of application servers and clients as well as default property values for each combination.

Interoperability of Version 4.0.x client with Version 5 (and later) application server

Ideally, all 4.0.x clients that use Version 5 or later application servers should be at Version 4.0.3 or later.

Version 5 and later application servers return the interoperability class types by default (`true`). This can cause interoperability problems for distributed clients at versions 4.0.1 or 4.0.2. In particular, problems can occur with collections and enumerations returned by Enterprise JavaBeans Version 1.1 finder methods.

Although it is strongly discouraged, you can set com.ibm.websphere.container.portable to `false` on a Version 5 and later application server. This causes the application server to return the old class types, providing interoperability with clients at Version 4.0.2 and earlier. This is discouraged because:
- The Version 5 application server instance would become non-J2EE 1.3 (and later) compliant with regard to handles, home interface handles, and EJBMetaData.
- EJB 1.x finder methods return collection and enumeration objects that do not originate from ejbportable.jar.
- Interoperability restrictions still exist with the property set to `false`.

- Version 5 and later client handles to entity beans and home interfaces do not work across domains for the server you set to `false`.

  If you would like to use updated Handle classes in EJB 2.x-compliant beans but have one of the older clients (versions 4.0.2 and earlier) installed, set the system property com.ibm.websphere.container.portable.finder to `false`. With this setting in place, the Version 5 and later application server uses the updated handles but returns the enumerations and collections that were used in the earlier clients.

Interoperability of client at Version 4.0.2 and earlier with Version 5 (and later) application server

Client at Version 4.0.2 and earlier, using this function	Application server at Version 5 and later, property true (default)	Application server at Version 5 and later, property false
EJBMetaData	Does not work	Works for 4.0.2 client
Handle to session bean	Does not work	Works
Handle to entity bean	Does not work	Does not work across cells
Enumeration returned by EJB 1.x finder method	Does not work	Works
Collection returned by EJB 1.x finder method	Does not work	Works
Handle to home interface	Does not work	Does not work across cells

If you would like to use updated Handle classes in EJB 2.x-compliant beans but have one of the older clients (versions 4.0.2 and earlier) installed, set the system property com.ibm.websphere.container.portable.finder to `false`. With this setting in place, the Version 5 and later server uses the new Handle classes but returns the older enumeration and collection classes.

Interoperability of client at Version 4.0.3 and later with Version 5 and later application server

Clients at Version 4.0.3 and later work well with Version 5 and later application servers. However, if you set the com.ibm.websphere.container.portable to `false`, client handles to entity beans and home interfaces do not work across domains for the server you set to `false`.

Client at Version 4.0.3 and later, using this function	Application server at Version 5 and later, property true (default)	Application server at Version 5 and later, property false
EJBMetaData	Works	Works
Handle to session bean	Works	Works
Handle to entity bean	Works	Does not work across cells
Enumeration returned by EJB 1.x finder method	Works	Works
Collection returned by EJB 1.x finder method	Works	Works
Handle to home interface	Works	Does not work across cells

Interoperability of Version 5 and later client with Version 4.0.x application server

Clients at Version 5 and later work well with Version 4.0.3 application servers if you set com.ibm.websphere.container.portable to `true`. Client handles to entity beans and home interfaces do not work across domains for any Version 4.0.3 server with com.ibm.websphere.container.portable at the default value, `false`. Version 5 client handles to application servers at Version 4.0.2 and earlier also have restrictions.

Client at Version 5 and later, using this function	Application server at Version 4.0.3, property true	Application server at Version 4.0.3, property false (default)	Application server at Version 4.0.2 or earlier
EJBMetaData	Works	Works	Works for 4.0.2 server only
Handle to session bean	Works	Works	Works
Handle to entity bean	Works	Does not work across domains	Does not work across domains
Enumeration returned by EJB 1.x finder method	Works	Works	Works
Collection returned by EJB 1.x finder method	Works	Works	Works
Handle to home interface	Works	Does not work across domains	Does not work across domains

Interoperability of zSeries Version 4.0.x client with Version 5 and later application server

The only valid configuration for container interoperability with zSeries Version 4.0.x clients is the default configuration for the Version 5 application server.

Interoperability of Version 5 and later client with zSeries Version 4.0.x application server

Version 5 clients should work with a zSeries Version 4.0.x application server with the correct interoperability fixes described in the zSeries documentation. The interoperability characteristics should be the same as for a Version 4.0.3 distributed application server with the property set to `true`.

Client at Version 5 and later, using this function	zSeries application server at Version 4.0.x
EJBMetaData	Works
Handle to session bean	Works
Handle to entity bean	Works
Enumeration returned by EJB 1.x finder method	Works
Collection returned by EJB 1.x finder method	Works
Handle to home interface	Works

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.

Each of the technology components have requirements that you can read about in more detail at the WS-I Organization Web site. For example, support for Universal Transformation Format (UTF)-16 encoding is required by WS-I Basic Profile. UTF-16 is a kind of Unicode encoding scheme using 16-bit values to store Universal Character Set (UCS) characters. UTF-8 is the most common encoding that is used on the Internet; UTF-16 encoding is typically used for Java and Windows product applications; and UTF-32 is used by various Linux and Unix systems. Unlike UTF-8, UTF-16 has issues with big-endian and little-endian, and often involves Byte Order Mark (BOM) to indicate the endian. BOM is mandatory for UTF-16 encoding and it can be used in UTF-8.

The following table summarizes some of the properties of each UTF:

Bytes	Encoding form
EF BB BF	UTF-8
FF FE	UTF-16, little-endian
FE FF	UTF-16, big-endian
00 00 FE FF	UTF-32, big-endian
FF FE 00 00	UTF-32, little-endian

BOM is written prior to the XML text, and it indicates to the parser how the XML is encoded. The XML declaration contains the encoding, for example: <?xml version=*xxx* encoding="utf-*xxx*"?>. BOM is used with the encoding to determine how to interpret the XML. Here is an example of a SOAP message and how BOM and UTF encoding are used:

```
POST http://www.whitemesa.net/soap12/add-test-rpc HTTP/1.1
Content-Type: application/soap+xml; charset=utf-16; action=""
SOAPAction:
Host: localhost: 8080
Content-Length: 562

0xFF0xFE<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2002/12/soap-envelope"
 xmlns:soapenc="http://www.w3.org/2002/12/soap-encoding
 xmlns:tns="http://whitemesa.net/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 from UTF-8 to UTF-16

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.

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");
  stub.setProperty(com.ibm.wsspi.webservices.Constants.XML_CHARSET,"UTF-16");
  stub.invokeMethod();
  ```

  In this code example, **com.ibm.wsspi.webservices.Constants.XML_CHARSET** = *"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 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.
- If you do not have a WSDL document for the service, do not expect the service to change, and you want to port the Apache SOAP client with minimal work, you can convert the code to use the JAX-RPC dynamic invocation interface (DII), which is similar to the Apache SOAP APIs. The DII APIs do not use WSDL or generated bindings.

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

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

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

   in JAX-RPC.

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

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

      in Apache SOAP.

      An instance of the Call object is created by

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

      in JAX-RPC.

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

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

      in Apache SOAP.

      The setTargetEndpointAddress method is used as a parameter to

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

      in JAX-RPC.

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

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

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

      in Apache SOAP.

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

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

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

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

in Apache SOAP.

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

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

in JAX-RPC.

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

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

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

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

Where

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

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

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

in Apache SOAP.

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

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

in JAX-RPC.

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

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

in Apache SOAP.

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

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

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

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

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

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

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

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

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

Migrating to Version 3 of the UDDI Registry

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

Use this topic to migrate to a Version 3 UDDI Registry that uses a DB2 database, from a Version 2 UDDI Registry (as defined in UDDI Registry Terminology).

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

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

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

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

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

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

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

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

1. Stop the UDDI Registry application that is running in your Version 5.x application server. This prevents further UDDI requests being directed to the UDDI Registry and ensures that no new data is published during the migration process.

2. Record information about the uddi.properties values being used. This file is located in the *DeploymentManager_install_dir*/config/cells/*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 from WebSphere Application Server Version 5.x to Version 6 (see Migration and coexistence overview). This results in a new directory tree for the migrated Version 6 application server.
4. Start the new migrated (Version 6) application server.
5. Create a new datasource for the Version 2 UDDI database. This is known as the UDDI migration datasource. Note that the JNDI name **must** be **datasources/uddimigration**. To complete this step see Setting up a UDDI migration datasource.
6. Follow all the instructions in Setting up a customized UDDI node in the *Administering applications and their environment* PDF book, including its subtopic relating to node initialization, to set up the UDDI Version 3 Registry and migrate the Version 2 data. 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 datasource
   - Deploy the UDDI registry application
   - Start the server
   - Configure and initialize the node. The UDDI Registry node initialization detects that there is a UDDI migration datasource, and migrates the Version 2 data as part of the UDDI node initialization processing.

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

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

   Should the following error appear:

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

   it means that an unexpected error has occurred during migration. The UDDI Registry node will not be activated. Check the error logs for the problem and, if it cannot be fixed contact your IBM representative for advice.
8. Once migration is complete the UDDI migration datasource may be removed, 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 DB2 JDBC Provider does not already exist, then create one, selecting the options DB2 Legacy CLI-based Type 2 JDBC Driver and Connection Pool datasource.
2. Create a datasource for the Version 2 UDDI Registry by following these steps:
   a. Expand **Resources** and **JDBC Providers**.
   b. 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.
   c. Select the JDBC provider created earlier.

d. Under **Additional Properties**, select **Data Sources** (*not* the Data Sources Version 4 option).
e. Click **New** to create a new datasource.
f. Fill in the details for the datasource as follows:

Name a suitable name, such as UDDI Datasource

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

Use this Data Source in container-managed persistence (CMP)
ensure the check box is cleared.

Description
a suitable description

Category
set to **uddi**

Data store helper class name
filled in for you as: com.ibm.websphere.rsadapter.DB2DataStoreHelper

Component-managed authentication alias
select the alias for the DB2 userid used to access UDDI Version 2 data, for example MyNode/UDDIAlias (refer to step 2 of Setting up a customized UDDI node in the *Administering applications and their environment* PDF book if you do not have such an alias)

Container-managed authentication alias
set to (none)

Mapping-configuration alias
set to DefaultPrincipalMapping

DB2 legacy CLI-based type 2 data source properties
set **Database name** to UDDI20, or the name given to your Version 2 UDDI DB2 database

Leave all other fields unchanged.

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.

Initializing the UDDI Registry node

Use this topic to initialize a UDDI Registry node after set up or migration.

You must have already set up a UDDI Registry node, either as a new node or to use for migrating a UDDI Registry Version 2 node.

The UDDI Registry node has various properties, some of which must be set before initializing the node. There are two categories of UDDI Registry node properties:

- Mandatory node properties. These properties must be set before the UDDI node can be initialized. You may set these properties as many times as you wish before initialization. However, once the UDDI node has been initialized, these properties will become read only for the lifetime of that UDDI node. It is very important to set these properties correctly.
- All other properties. These properties may be set before, and after, initialization.

Configure these properties and initialize the node using the UDDI administrative console or JMX management interface.

1. Click **UDDI** → **UDDI Nodes** > *UDDI_node_id* to display the properties page for the UDDI Registry node.
2. Set the mandatory node properties to suitable, and valid, values. These properties are indicated by the presence of a '*' next to the input field. The properties are listed below; more information on each property is given in the context help of the administrative console.

   UDDI node ID
   This must be a text string beginning with 'uddi:' that is unique to this UDDI node. The default value may be sufficient, but if you accept it you should ensure that it is unique.

   UDDI node description
   This is a text string describing the node.

   Root key generator
   This must be a text string beginning with 'uddi:' that is unique to this UDDI node. The default value may be sufficient but may contain text, such as 'keyspace_id', that you should modify to match your system. If you accept the default value, ensure that it is unique for this UDDI node.

   Prefix for generated discoveryURLs
   This should be a valid URL.
3. If you are migrating from Version 2 of the UDDI Registry, use the table below to perform the following steps:
   - Set any properties from uddi.properties that **must** remain the same as Version 2.
   - Set any properties from uddi.properties that you would like to keep the same (such as dbMaxResultCount).

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

4. Set any other properties, such as policy values, that you wish to change from the default settings (or these can be changed at a later time).
5. Once the properties have been set to appropriate values, click **Apply** to save your changes. It is important to save the changes before proceeding to the initialize step.
6. Initialize the UDDI node by clicking **Initialize**, at the top of the pane. If you are migrating from Version 2 of the UDDI Registry, the Version 2 data is migrated now. The initialization may take some time to complete.

If the node has been migrated from a previous version, return to “Migrating to Version 3 of the UDDI Registry” on page 57 to verify that the migration was successful. If you have created a new node, you are now ready to use the UDDI Registry.

Using a remote database for the UDDI Registry

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

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

Remote DB2

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

Networked Cloudscape

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

For details of how to set up Cloudscape for multiple connections see Configuring Cloudscape Version 5.1.60x in the *Administering applications and their environment* PDF book.

Note: Embedded Cloudscape is not supported for this configuration.

Data access resources

Migrating a Version 4.0 data access application to Version 6.0

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

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, however, you cannot 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.

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.
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 the documentation for Rational Application Developer, or the documentation for WebSphere Studio Application Developer Integration Edition.
5. Add any necessary resource references.
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 anassembly tool.
2. Create a new Application by selecting **File > New > Application**.
3. Add each of your EJB modules.
   a. Right-click **EJB Modules** and select **Import**.
   b. Navigate to your converted EJB module and click **Open**.
   c. Click **OK**.
4. Add each of your Web modules.
   a. Right-click **Web Modules** and select **Import**.
   b. Navigate to your converted Web module and click **Open**.
   c. Fill in a **Context root** and click **OK**.
5. Identify any other application properties. Click **Help** for a description of the settings.
6. Save the EAR file.

Installing the Application on WebSphere Application Server

1. Create a JDBC provider and a data source object..
2. Install the application and bind the resource references to the data source that you created.

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

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

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

This change affects all applications that contain the following methods:

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

Symptoms of the problem include:
- Session hang
- Session timeout
- Running out of connections

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

Explanation of the underlying problem

For J2EE 1.2 modules using Version 4 data sources, WebSphere Application Server issues non-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.

Mail, URLs, and other J2EE resources

Mail migration tip

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

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

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

Security

Interoperability issues for security

To have interoperability of Security Authentication Service (SAS) between C++ and WebSphere Application Server, use the Common Secure Interoperability Version 2 (CSIv2) authentication protocol over Remote Method Invocation over the Internet Inter-ORB Protocol (RMI-IIOP).

To have interoperability of SAS between WebSphere Application Server and WebSphere Application Server for z/OS use the zSAS authentication protocol over RMI-IIOP.

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

You can achieve interoperability between C++ CORBA clients and WebSphere Application Server using the z/OS Secure Authentication Services (z/SAS) protocols. z/SAS supports many of the same functions as Common Secure Interoperability Version 2 (CSIv2), only z/SAS uses a proprietary architecture. z/SAS supports three types of authentication:

- User ID and password authentication
- User ID and password authentication over SSL
- SSL client certificate authentication

Security authentication from non-Java based C++ client to enterprise beans. WebSphere Application Server supports security in the CORBA C++ client to access protected enterprise beans. If configured, C++ CORBA clients can access protected enterprise bean methods using client certificate to achieve mutual authentication on WebSphere Application Server applications.

To support the C++ CORBA client in accessing protected enterprise beans:

- Create an environment file for the client, such as current.env. Set the variables listed below (security_sslKeyring, client_protocol_user, client_protocol_password) in the file.
- Point to the environment file using the fully qualified path name through the environment variable WAS_CONFIG_FILE. For example, in the test shell script `test.sh`, export:

  ```
  /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

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

Interoperating with previous product versions

IBM WebSphere Application Server, Version 6.0.x interoperates with the previous product versions such as Version 5.x. Interoperability is achieved using the zSAS security mechanism for localOS and SAF-based authorization.

1. If SSL is configured on a previous product version, your servers must have a basis to establish trust. Using Resource Access Control Facility (RACF), your system can check to ensure that the intermediate server can be trusted (to confer this level of trust, CBIND authorization is granted by administrators to RACF user IDs that run secure system code). System SSL repertoires use a System Authorization Facility (SAF) keyring to retrieve the personal certificate and trust stores. You must connect the trust basis for the server certificates (on the default setup the certificate authority certificate) of the previous version server into the keyring of the WebSphere Application Server for z/OS Version 6.0.x server.
2. Extract and add server certificates into the server key ring file of the previous version.
   a. Open the server key ring file using the key management utility (iKeyman) and extract the server certificate to a file.
   b. Open the server key ring of the previous product version, using the key management utility and add the certificate extracted from WebSphere Application Server Version 6.0.x.
3. Extract and add server certificates into the server key ring file of the previous version.
   a. Open the server key ring file using the key management utility (iKeyman) and extract the server certificate to a file.
   b. Open the server key ring of the previous product version, using the key management utility and add the certificate extracted from the product.
4. Extract and add trust certificates into the trust key ring file of the previous product version.
   a. Open the trust key ring file using the key management utility and extract the trust certificate to a file.
   b. Open the trust key ring file of the previous product version using the key management utility and add the certificate extracted from the product.
5. If single signon (SSO) is enabled, export keys from the product and import them into the previous product version.
6. Verify that the application uses the correct JNDI name. In WebSphere Application Server Version 6.0.x, the enterprise beans are registered with long JNDI names like, `(top)/nodes/`*`node_name`*`/servers/`*`server_name`*`/HelloHome`. Whereas in previous releases, enterprise

beans are registered under a root like, (top)/HelloHome. Therefore, EJB applications from previous versions perform a lookup on the Version 6.0.x enterprise beans.

You can also create EJB name bindings that are compatible with the previous version. To create an EJB name binding at the root Version 6.0.x, start the administrative console and click **Environment** > **Naming** > **Naming Space Bindings** > **New** > **EJB** > **Next**. Complete all the fields and enter a short name (for example, -HelloHome) as the JNDI Name. Click **Next** and **Finish**.

7. Stop and restart all the servers.
8. Make sure that the correct naming bootstrap port is used to perform naming lookup. In previous product versions, the naming bootstrap port is **900**. In Version 6.0.x, the bootstrap port is **2809**.

Migrating security configurations from previous releases

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

- Before migrating your configurations, verify that the administrative server of the previous release is running.
- If security is enabled in the previous release, obtain the server ID and password of the previous release. This information is needed to log onto the administrative server of the previous release during migration.
- You can optionally disable security in the previous release before migrating the installation. There is no logon required during the installation.

1. Start the First steps wizard by launching the firststeps.bat or firststeps.sh file. The first steps file is located in the following directory:
   - *./install_root*/profiles/*profile_name*/firststeps/firststeps.sh
2. On the First steps wizard panel, click **Migration wizard**.
3. Follow the instructions provided in the First steps wizard to complete the migration.

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

This task is for migrating an installation.

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

If you upgrade from WebSphere Application Server, Version 5.x or 4.0.x to WebSphere Application Server, Version 6.0.x, the data that is associated with Version 5.x or 4.0.x trust associations is not automatically migrated to Version 6.0.x. To migrate trust associations, see “Migrating trust association interceptors” on page 70.

Migrating custom user registries

Before you perform this task, it is assumed that you already have a custom user registry implemented and working with WebSphere Application Server Version 6. The custom registry interface is the UserRegistry interface.

In WebSphere Application Server Version 6.0.x, 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, datasources) 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 EJB method getCallerPrincipal( ) and the servlet methods getUserPrincipal( ) and getRemoteUser( ) return the security name instead of the display name. However, if you need the display names to return, set the WAS_UseDisplayName property to **true**. See the getUserDisplayName method description or the Javadoc, for more information.

If the migration tool was used to migrate the WebSphere Application Server Version 5 configuration to WebSphere Application Server Version 6.0.x, be aware that this migration does not involve any changes to your existing code. Because the WebSphere Application Server Version 5 custom registry works in WebSphere Application Server Version 6.0.x 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. Consider that the migration tool might not copy your implementation files from Version 4 to Version 6.0.x. You might have to copy them to the class path in the Version 6 setup (preferably to the `classes` subdirectoy). If you are using the WebSphere Application Server Network Deployment version, copy the files to the cell and to each of the nodes class paths.

In WebSphere Application Server Version 6.0.x, a case insensitive authorization can occur when using the custom registry. This authorization is in effect only on the authorization check. This function is useful in cases where your custom registry returns inconsistent (in terms of case) results for user and group unique IDs.

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

Before proceeding, look at the UserRegistry interface. See Developing custom user registries in the *Securing applications and their environment* PDF book for a description of each of these methods in detail.

The following steps go through in detail all the changes required to move your WebSphere Application Server Version 5 custom user registry to the WebSphere Application Server Version 6.0.x custom user registry. The steps are very simple and involve minimal code changes. The sample implementation file is used as an example when describing some of the steps.

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

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

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

   Having a certificate chain gives you the flexibility to act on the chain instead of one certificate. If you are only interested in the first certificate just take the first certificate in the chain before processing. In WebSphere Application Server Version 6, the mapCertificate method is called to map the user in a certificate to a valid user in the registry, when certificates are used for authentication by the Web or the Java clients (transport layer certificates, Identity Assertion certificates).
4. Remove the getUsers() method.
5. Change the signature of the getUsers(String) method to return a Result object and accept an additional parameter (int). Change:

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

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

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

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

to

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

In WebSphere Application Server Version 6, this method is not called directly by the WebSphere Application Server Security component. However, other components of the WebSphere Application Server like the WebSphere Business Integration Server Foundation Process Choreographer use this method when staff assignments are modeled using groups. Because this already is implemented in WebSphere Application Server Version 6.0.x, it is recommended that you change the implementation similar to the `getUsers` method as explained in step 5.

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

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

to

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

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

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

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

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

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

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

Type the previous lines as one continuous line.

12. Copy the implementation classes to the product class path. The *%install_root%*/lib/ext directory is the preferred location. If you are using the Network Deployment product, make sure that you copy these files to the cell and all the nodes. Without the files 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 Configuring custom user registries article in the *Securing applications and their environment* PDF book to set up the custom registry including the IgnoreCase flag. Make sure that you add the WAS_UseDisplayName properties, if required.

Migrates to a WebSphere Application Server Version 6.0.x custom registry.

This step is required to migrate a custom registry from WebSphere Application Server Version 5 to WebSphere Application Server Version 6.0.x.

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

Migrating trust association interceptors

Changes to the product-provided trust association interceptors

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

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

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

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

Migrating product-provided trust association interceptors

The properties located in the webseal.properties and trustedserver.properties files are not migrated from previous versions of the WebSphere Application Server. You must migrate the appropriate properties to WebSphere Application Server Version 6 using the trust association panels in the administrative console..

Changes to the custom trust association interceptors

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

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

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

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

Method 1:

Add the `com.ibm.websphere.security.trustassociation.initStatus` property in the trust association interceptor custom properties. Set the property to the value that indicates that the interceptor is successfully initialized. All of the other possible values imply failure. In case of failure, the corresponding trust association interceptor is not used.

Method 2:

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

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

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

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

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

Method 1:

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

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

Method 2:

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

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

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

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

Migrating custom trust association interceptors

The trust associations from previous versions of WebSphere Application Server are not automatically migrated to WebSphere Application Server Version 6.0.x. Users can manually migrate these trust associations using the following steps:

1. Recompile the implementation file, if necessary.

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

   To recompile the implementation file, type the following:

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

   Note: The previous line of code was broken into two lines due to the width of the page. Type the code as one continuous line.
2. Copy the custom trust association interceptor class files to a location in your product class path. It is suggested that you copy these class files into the `%WAS_HOME%/lib/ext` directory.
3. Start the WebSphere Application Server.
4. Enable security to use the trust association interceptor. *The properties located in your custom trust association properties file and in the `trustedserver.properties` file are not migrated from previous versions of WebSphere Application Server. You must migrate the appropriate properties to WebSphere Application Server Version 6 using the trust association panels in the administrative console.*

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

Note: 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 wish 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).

WebSphere Application Server fully supports the Java Authentication and Authorization Service (JAAS) as programmatic login application programming interfaces (API).

This document outlines the deprecated CORBA programmatic login APIs and the alternatives provided by JAAS. The following are the deprecated CORBA programmatic login APIs and are not supported on WebSphere Application Server for z/OS:

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

The `sampleApp` is not included in Version 6.

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

  The `sampleApp` is not included in Version 6.
- **com.ibm.IExtendedSecurity._LoginHelper**.

  This API is not included in Version 6.
- **org.omg.SecurityLevel2.Credentials**.

  This API is included with the product, but is not to be used with z/OS.

The APIs provided in WebSphere Application Server Version 6.0.x are a combination of standard JAAS APIs and a product implementation of standard JAAS interfaces.

The supported APIs provided in WebSphere Application Server for z/OS Version 6 are a combination of standard JAAS APIs and product implementation of standard JAAS interfaces (also 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 when the application pushes basic authentication data (user ID, password, and security realm) or token data to product LoginModules. This API is recommended for server-side login.

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

    Note: If this API is used on the server side, the server is blocked for input.
  - javax.security.auth.callback.Callback interface:

    javax.security.auth.callback.NameCallback
    Provided by JAAS to pass the user name to the LoginModules interface.

    javax.security.auth.callback.PasswordCallback
    Provided by JAAS to pass the password to the LoginModules interface.

    com.ibm.websphere.security.auth.callback.WSCredTokenCallbackImpl
    Provided by the product to perform a token-based login. With this API, an application can pass a token-byte array to the LoginModules interface.
  - **javax.security.auth.spi.LoginModule interface**

    WebSphere Application Server provides LoginModules implementation for client and server-side login.
- 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

  Note: An application must invoke the WSSubject.doAs() method for J2EE resources to be accessed using the subject generated by an explicit invocation of a WebSphere login module.

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

com.ibm.websphere.security.auth.WSPrincipal
An authenticated principal, that is created and stored in a Subject that is authenticated by the WebSphere LoginModules interface.

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 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 WSPrincipal.
java.util.Set ps =
s.getPrincipals(com.ibm.websphere.security.auth.WSPrincipal.class);
java.util.Iterator it = ps.iterator();
while (it.hasNext()) {
com.ibm.websphere.security.auth.WSPrincipal p =
(com.ibm.websphere.security.auth.WSPrincipal) it.next();
System.out.println("Principal: " + p.getName());
}
} catch (javax.security.auth.login.LoginException e) {
System.err.println("ERROR: login failed with exception: " + e.getMessage());
e.printStackTrace();

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

Migrating from the CustomLoginServlet class to servlet filters

The CustomLoginServlet class was deprecated in 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 that displays when retrieving the user ID and password information. You also can specify an error page that displays when authentication fails.

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

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

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

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

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

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

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 on. In this case, the security policy of the previously installed applications would not exist in the JACC provider to make the access decisions. You can reinstall all of the applications once JACC is enabled. The wsadmin scripting tool can be used to propagate information to the JACC provider independent of the application install 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. The wsadmin tool can be used to invoke this method at the deployment manager level.

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.

The appNames 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 application(s) to the Tivoli Access Manager JACC provider, please create or import the users and groups that are in the application(s) to Tivoli Access Manager.
- Depending on the application or the number of applications propagated you might have to increase the request time-out period either in the soap.client.props (if using SOAP) or the sas.client.props (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.
2. Restart the server.
3. Enter the following commands:

```
// use the SecurityAdmin Mbean at the Deployment Manager or the unmanaged base application server
wsadmin -user serverID -password serverPWD
set secadm [lindex [$AdminControl queryNames type=SecurityAdmin,*] 0]

// 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
```

Enabling embedded Tivoli Access Manager

Embedded Tivoli Access Manager is not enabled by default but needs to be configured 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 is the user registry containing the user and group information for both Tivoli Access Manager and WebSphere Application Server.
- A Tivoli Access Manager Version 5.1 domain exists and is configured to use the user registry. For details on the installation and configuration of Tivoli Access Manager refer to the: *Tivoli Access Manager Base installation Guide* and the *Tivoli Access Manager Base Administrator's Guide* available from http://publib.boulder.ibm.com/tividd/td/tdprodlist.html.
- WebSphere Application Server Version 6.0.x is installed either in a single server model or as a network deployment.

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

1. Create the security administrative user.
2. Configure the Tivoli Access Manager Java Authorization Contract for Containers (JACC) provider.
3. Enable WebSphere Application Server security. When you are using Tivoli Access Manager you must configure LDAP as the user registry.
4. Enable the Tivoli Access Manager JACC provider.

Migrating Java 2 security policy

Previous WebSphere Application Server releases

WebSphere Application Server uses the Java 2 security manager in the server run time to prevent enterprise applications from calling the `System.exit()` and the `System.setSecurityManager()` methods. These two Java 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 an undesirable operation for an application server.

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

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

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

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

Only two permissions are supported:

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

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

What changed

Java 2 Security is fully supported in WebSphere Application Server Version 6.0.x, which means all permissions are enforced. The default Java 2 security policy for enterprise application is the recommended permission set defined by the Java 2 Platform, Enterprise Edition (J2EE) Version 1.4 specification. Refer to the *install_root*`/profiles/`*profile_name*`/config/cells/`*cell_name*`/nodes/`*node_name*`/app.policy` file for the default Java 2 security policy granted to enterprise applications. This is a much more stringent policy compared to previous releases.

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

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

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 **Enforce Java 2 Security** option on the Global security settings 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). It contains both system permissions (permissions granted to the Java Virtual Machine (JVM) and the product server run time) and enterprise application permissions. Migrate the Java 2 security policy of the enterprise application to 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 is deprecated; a flag in the WebSphere configuration application programming interface (API) is used to control whether to enabled Java 2 security. Enable this option through the administrative console.
- **was.home**. Expanded to the installation directory of the WebSphere Application Server (action might be required). This is deprecated; superseded by `${user.install.root}` and `${was.install.root}` properties. If the directory contains instance specific data then `${user.install.root}` is used; otherwise `${was.install.root}` is used. Use these properties interchangeably for the WebSphere Application Server or the Network Deployment environments. See the steps for migrating Java 2 security policy.

Migrating the Java 2 Security Policy

There is no easy way of migrating the Java policy file to WebSphere Application Server Version 6.0.x automatically because there is a mixture of system permissions and application permissions in the same policy file. Manually copy the Java 2 security policy for enterprise applications to a `was.policy` or `app.policy` file. However, migrating the Java 2 security policy to a `was.policy` file is preferable because symbols or relative codebase is used instead of absolute codebase. There are many advantages to this

process. The permissions defined in the was.policy file should only be granted to the specific enterprise application, while permissions in the app.policy file apply to all the enterprise applications running on the node where the app.policy file belongs.

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

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

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

1. Ensure that Java 2 security is disabled on the application server.
2. Create a new was.policy file (if one is not present) or update the was.policy for migrated applications in the configuration repository in (profiles/*profile_name*config/cells/*cell_name*/applications/app.ear/deployments/app/META-INF/was.policy) with the following contents:

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

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

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

Naming and directory

JNDI interoperability considerations

This section explains considerations to take into account when interoperating with WebSphere Application Server V4.0 and with non-WebSphere Application Server JNDI clients. Also, the way resources from MQSeries must be bound to the name space changed after V4.0 and is described below.

Interoperability with WebSphere Application Server V4.0

- **EJB clients running on WebSphere Application Server V4.0 accessing EJB applications running on WebSphere Application Server V5 or V6**

  Applications migrated from previous versions of WebSphere Application Server may still have clients still running in a previous release. The default initial JNDI context for EJB clients running on previous versions of WebSphere Application Server is the cell persistent root (legacy root). The home for an enterprise bean deployed in version 5 or 6 is bound to its server's server root context. In order for the EJB lookup name for down-level clients to remain unchanged, configure a binding for the EJB home under the cell persistent root.

- **EJB clients running on WebSphere Application Server V5 or V6 accessing EJB applications running on WebSphere Application Server V4.0 servers**

  The default initial context for a WebSphere Application Server V4.0 server is the correct initial context. Simply look up the JNDI name under which the EJB home is bound.

  Note: To enable WebSphere Application Server V5 or V6 clients to access version 4.x servers, the down-level installations must have e-fix PQ60074 installed.

EJB clients running in an environment other than WebSphere Application Server accessing EJB applications running on WebSphere Application Server V5 or V6 servers

When an EJB application running in WebSphere Application Server V5 or V6 is accessed by a non-WebSphere Application Server EJB client, the JNDI initial context factory is presumed to be a non-WebSphere Application Server implementation. In this case, the default initial context will be the cell root. If the JNDI service provider being used supports CORBA object URLs, the corbaname format can be used to look up the EJB home. The construction of the stringified name depends on whether the object is installed on a single server or cluster, as shown below.

- **Single server**

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

  According to the URL above, the bootstrap host and port are myHost and 2809, and the enterprise bean is installed in a server **server1** in node **node1** and bound in that server under the name **myEJB**.

- **Server cluster**

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

  According to the URL above, the bootstrap host and port are **myHost** and **2809**, and the enterprise bean is installed in a server cluster named **myCluster** and bound in that cluster under the name **myEJB**.

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

The above lookup will also work if the bootstrap host and port belongs to a member of the cluster itself. To avoid a single point of failure, the bootstrap server host and port for each cluster member could be listed in the URL as follows:

```
initialContext.lookup(
    "corbaname:iiop:host1:9810,:host2:9810#cell/clusters/myCluster/myEJB");
```

The name prefix **cell/clusters/myCluster/** is not necessary if boostrapping 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.

Learn about WebSphere programming extensions

Use this section as a starting point to investigate the WebSphere programming model extensions for enhancing your application development and deployment.

Application profiling

Migrating Version 5 Application Profiles to Version 6

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

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

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

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

This situation only happens when:

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

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

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

Therefore it is recommended that you migrate any application you might have configured with application profiling in Version 5. Application profiles migration only requires you to re-configure your applications in the Version 6 Application Server Toolkit (AST).

Application profiling interoperability

The effect of 5.x Compatibility Mode

Application profiling supports *forward* compatibility. Application profiles created in previous versions of WebSphere Application Server (Enterprise Edition 5.0 or WebSphere Business Integration Server Foundation 5.1) can only run in WebSphere Application Server Version 6 if the Application Profiling 5.x Compatibility Mode attribute is turned on. If the 5.x Compatibility Mode attribute is off, Version 5 application profiles might display unexpected behavior.

Similarly, application profiles that you create using WebSphere Application Server Version 6 are not compatible with Version 5 or earlier versions. Even applications configured with application profiles run on Version 6 servers with the Application Profiling 5.x Compatibility Mode attribute turned on cannot interact with applications configured with profiles run on Version 5 servers.

Note: If you select the 5.x Compatibility Mode attribute on the Application Profile Service'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 6.0 client to interact with applications run under the Application Profiling 5.x Compatibility Mode, you must set the *appprofileCompatibility* system property to **true** in the client process. You can do this by specifying the *-CCDappprofileCompatibility=true* option when invoking the *launchClient* command.

Considerations for a clustered environment

In a clustered environment with mixed WebSphere Application Server product versions and mixed platforms, applications configured with application profiles might exhibit unexpected behavior because previous versions of server members cannot support the application profiling of Version 6.

If a clustered environment contains both Version 5.x and 6.0 server members, and if any applications are configured with application profiles, the Application Profiling 5.x Compatibility Mode attribute must be turned on in Version 6 server members. Still, this cluster can only support Version 5 application profiling behavior. To support applications configured with Version 6 application profiles in a cluster environment, all server members in the cluster must be at the Version 6 level.

WebSphere Application Server Enterprise Edition Version 5.0.2

If you use WebSphere Application Server Enterprise Edition 5.0.2, you must apply WebSphere Application Server Version 5 service pack 7 or later service pack to enable Application Profiling interoperability.

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, .

1. Install the Version 6 product.
2. Use the Version 6 Profile creation wizard to create one or more profiles for a deployment manager, a managed node, or a stand-alone Application Server.
3. Start the **First steps** console.
4. Select the **Migration wizard** on the **First steps** console.
5. Use the to migrate the previous release to the Version 6 product.

Dynamic cache

Migrating V6.0 servers from multi-broker replication domains to data replication domains

Use this task to migrate multi-broker replication domains to data replication domains. Multi-broker domains 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 V6.0 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.

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, create an empty data replication domain. Click **Environment > Replication domains > New** in the administrative console.
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.

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 V6.0 application servers use the two types of replication domains:

	V5.x application servers using replication domains	**V6.0 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.
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.

	V5.x application servers using replication domains	V6.0 application servers using replication domains
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.
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.

Internationalization

Migrating internationalized applications

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

In Version 4, the internationalization service did not provide internationalization deployment descriptor policy information to direct how the service manages internationalization context across the various application components. Rather, the service employed the implicit client-side internationalization (CSI) and server-side internationalization (SSI) policies, which dictated how the service managed context according to the type of Java 2 Platform, Enterprise Edition (J2EE) container hosting a component. For details, refer to the combined information center for WebSphere Application Server Version 4 (www7b.software.ibm.com/wsdd/WASInfoCenter/infocenter). Briefly, all server components in Version 4 are SSI, and all EJB client applications are CSI.

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

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

Scheduler

Interoperating with schedulers

Schedulers support forward compatibility. Tasks created in previous versions of WebSphere Application Server Enterprise Edition 5.0 or WebSphere Business Integration Server Foundation 5.1 continue to run in WebSphere Application Server, Version 6 schedulers. Tasks that you create using Version 6 are not compatible with product schedulers from Version 5 or earlier. Version 5 schedulers do not run any Version 6 tasks.

All schedulers that are configured to use the same database and tables are considered a clustered scheduler. To guarantee that your tasks will run correctly, all servers in a scheduler cluster must be at the same version. If the servers are at different versions, tasks created with a Version 6 scheduler may not run. If a mixed-Version environment is required for a short period of time, then all scheduler poll daemons should be stopped on all Version 5 servers to allow a Version 6 server to run all tasks. This action allows the Version 6 schedulers to obtain leases and run tasks that have been created with a Version 6 scheduler.

Running tasks created with schedulers prior to Version 5.0.2 is not supported. See the topic, ″Interoperating with the Scheduler service,″ in the WebSphere Application Server Enterprise Edition Version 5.0.2 information center for details on how to migrate these tasks to a more recent version. See the Information Center Library (ww.ibm.com/software/webservers/appserv/infocenter.html) to access the Version 5.0.2 information center.

Chapter 4. Migrating product configurations

The migration utilities in V6.0.1 support migration from WebSphere Application Server for z/OS V5.x. Use this topic as a starting point for the planning information, Customization Dialog walkthroughs, and V5.x to V6.0.1 migration explanations for base Application Server nodes, deployment managers, and federated nodes.

1. Migrate a base Application Server node to V6.0.1. See “Base Application Server node migrations” on page 92 for more information.
2. Migrate a deployment manager to V6.0.1. See “Network Deployment migrations” on page 102 for more information.
3. Migrate a federated node to V6.0.1. See “Managed (federated) node migrations” on page 111 for more information.

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

Planning to migrate WebSphere Application Server for z/OS

Use this task to prepare to migrate WebSphere Application Server for z/OS to Version 6.0.1.

A new set of migration utilities is available in V6.0.1. To learn more about them, see “Overview of the V6.0.1 migration process.” When you are getting ready to migrate to V6.0.1, see “Base Application Server node migrations” on page 92, “Network Deployment migrations” on page 102, or “Managed (federated) node migrations” on page 111 for detailed planning and migration information.

1. See the *Installing your application serving environment* PDF book. This document will give you a basic understanding of what is involved with installing the product.
2. Install the V6.0.1 product.
3. To prepare your environment for migration, see Preparing the base operating system. You should also plan for any new features that you may want to include.
4. Review Ensuring problem avoidance in the *Installing your application serving environment* PDF book, making sure that each task has been completed before you begin.
5. Review “Migrating and coexisting” on page 17 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.
6. 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 in planning for the HFS structure.

When you have finished planning and are ready to begin migrating, continue to Chapter 4, “Migrating product configurations.”

Overview of the V6.0.1 migration process

Overview of the migration process

WebSphere Application Server for z/OS Version 6.0.1 requires that you migrate your V5.x configuration up to the V6.0.1 level. **You cannot simply point to the new V6.0.1 data sets and restart your servers.**

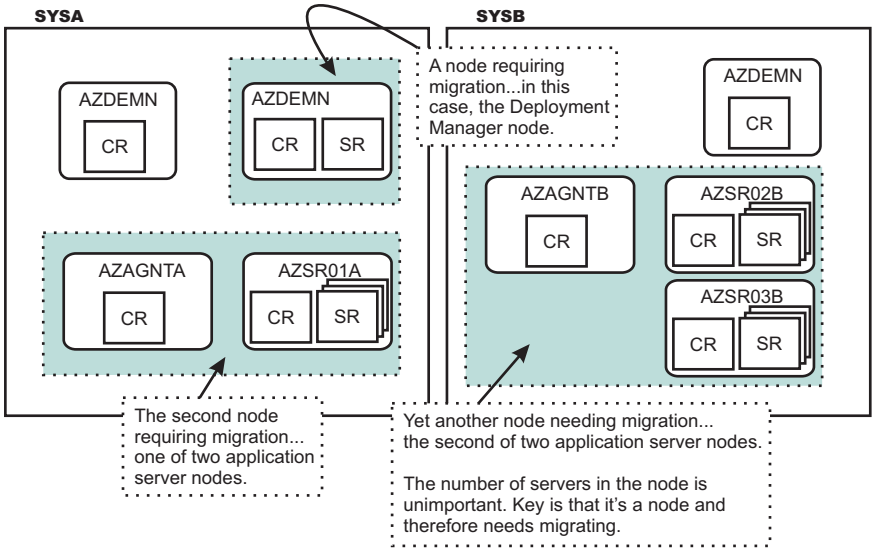
High-level description of the migration process

Install the SMP/E code for WebSphere Application Server for z/OS V6.0.1. Use the Customization Dialog walkthroughs to create the migration utilities you need to perform the migration.

When you run these jobs a new configuration is created — separate from your existing V5.x configuration — that is an exact copy in every way except that it will be at the V6.0.1 level.

Migration is a node-by-node activity

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



With a base 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 migration activities for base Application Server node" on page 101 for a checklist of activities for migrating a base Application Server node.

What do the migration utilities do?

The migration utilities serve the following purposes:

BBOWMG1B (base Application Server migrations), BBOWMG1F (federated node 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 is optional for configurations that do not use XA connectors.

BBOWMG2B (base 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 is optional for configurations that do not use XA connectors.

BBOWMBMT (base Application Server migrations), BBOWMDMT (deployment manager migrations), BBOWMMMT (federated node migrations)
Optional: Creates an HFS and mount point for the V6.0.1 configuration root. If you want to use an existing HFS to contain the V6.0.1 configuration, you must manually create the mount point specified during the migration dialogs rather than run this job. In either case, the HFS and mount point must be created prior to proceeding with the migration.

BBOWMG3B (base Application Server migrations), BBOWMG3D (deployment manager migrations), BBOWMG3F (federated node migrations)
Performs the migration of the node from V5.x to V6.0.1.

BBOMBCP (base 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 V6.0.1 configuration make use of different JCL start procedure names, this utility will update the new V6.0.1 configuration, substituting your new JCL names in place of the names that existed in your original V5.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 will copy the contents of your present V5.x configuration HFS into a new, separate configuration HFS, changing them along the way as needed to support V6.0.1:

Will my existing configuration be lost during migration?

During the migration the original V5.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 multi-node configuration it is possible to have the other nodes still running. But any node that you wish 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 V6.0.1 copy of the 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. V5.1 can coexist with V6.0.1 in the same cell and on the same LPAR. When migrating from V5.0.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. V5.0.x and V6.0.1 nodes cannot exist in the same cell on the same LPAR.

Can my newly-migrated V6.0.1 deployment manager still ″talk″ to V5.x nodes?

Yes. A deployment manager migrated to the Version 6.0.1 level of code can manage a V5.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 V6.0.1, a new copy of the ″master configuration″ is created. The old copy of the ″master configuration″ (the V5.x copy) still exists. But when the V6.0.1 deployment manager makes changes to the configuration, it makes it to the new V6.0.1 copy of the master configuration. So while it is possible to use to the V5.x copy of the code, any changes made in V6.0.1 will not be seen when the older code is restored.
- A V5.x deployment manager has no ability to manage a Version 6.0.1 node.

Is there a sequence to performing a multi-node migration?

Yes, there is:

1. Always migrate the deployment manager first.
2. V5.0.x Application Servers that reside on the same node as the deployment manager must be migrated next. V5.1 and V6.0.1 nodes can coexist in the same cell and on the same LPAR, so any V5.1 nodes do not need to be migrated immediately. See “Coexistence support” on page 121 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 V6.0.1 coexist with other cells at V5.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 may have the same cell short name.
- V5.0.x cannot coexist with V6.0.1 in the same cell on the same LPAR. If you have multiple V5.0.x nodes on the same LPAR, all nodes must be migrated to V6.0.1 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 HFS mount point and separate JCL procedures.

Base Application Server node migrations

The migration documentation for a base Application Server node includes planning information, a Customization Dialog walkthrough, a detailed V6.0.1 migration explanation, and a very high-level migration checklist. Select the appropriate link for information about how to migrate to a V6.0.1 base Application Server node:

- “Preparing to migrate a base Application Server node to V6.0.1”
- “Customization Dialog walkthrough for migrating a stand-alone Application Server node” on page 95
- “Migrating a base Application Server node” on page 99
- “Checklist of migration activities for base Application Server node” on page 101

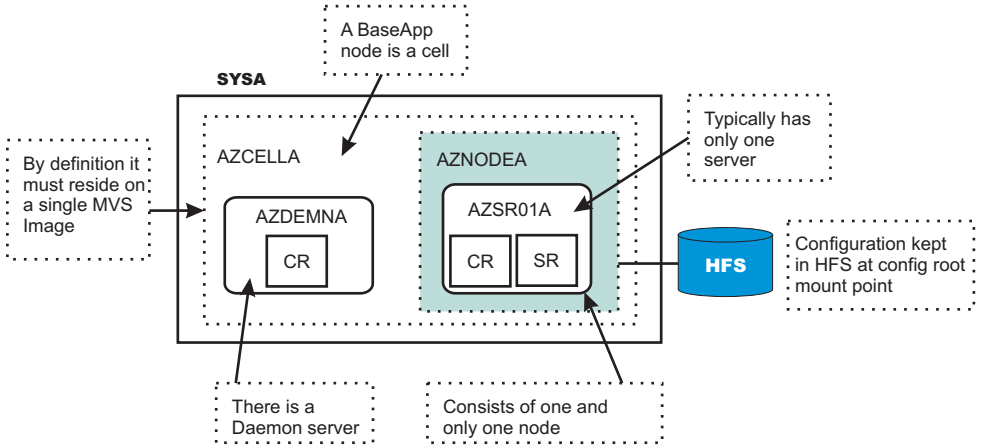
You can use the migration utilities to migrate a Version 5.x base Application Server node to WebSphere Application Server for z/OS Version 6.0.1.

Return to Chapter 4, “Migrating product configurations,” on page 89 to continue.

Preparing to migrate a base Application Server node to V6.0.1

Read the following information carefully to be sure that you are ready to migrate to V6.0.1.

Migrating a base Application Server node is relatively simple because it consists of a single server and a daemon:



You may have configured multiple Application Servers inside a base 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 6.0.1 supplies five migration utilities in the data set that you specify during the ″Allocate target data sets″ step in the Customization Dialog (see “Overview of the V6.0.1 migration process” on page 89 for a description of each utility):

- BBOWMG1B
- BBOWMG2B
- BBOWMBMT
- BBOWMG3B
- BBOMBCP

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

Inventory your existing configuration

WebSphere Application Server for z/OS V5.x configuration general information:

Cell short name:	
Cell long name:	
Sysplex name:	
Location of original V5.x configuration datasets:	
WebSphere Application Server V5.x home directory (configuration HFS):	

WebSphere Application Server for z/OS V5.x base Application Server node:

Configuration mount point:	
Configuration HFS data set:	
Home directory:	default: /AppServer your value: _____
WebSphere Application Server SMP/E home:	default: /usr/lpp/zWebSphere/V5R0M2 your value: _____

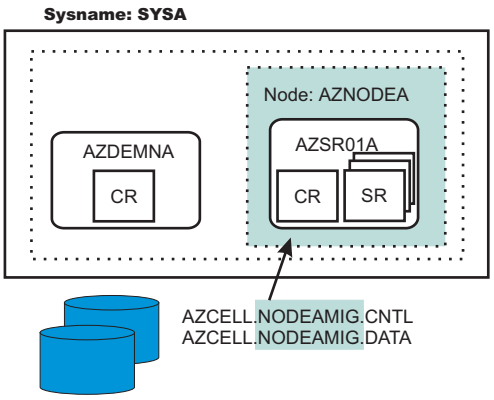
Controller JCL start proc:	
Servant JCL start proc:	
Daemon JCL start:	

WebSphere Application Server for z/OS V6.0.1 installation information:

SMP/E data set HLQ:	
SMP/E HFS Mount Point:	
WebSphere Application Server V6.0.1 product directory (SMP/E target for V6.0.1):	
WebSphere Application Server V6.0.1 in LPA/LNKLST?:	__ No __ Yes

Produce skeleton migration CNTL/DATA for 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 Dialogs for V6.0.1 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, on most of the panels it will be necessary to supply information related to your configuration. 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 some key information about your new V6.0.1 configuration (for example, mount points and paths), 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 (for example: settings and applications) from your existing V5.x configuration HFS.

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 walkthrough for migrating a stand-alone Application Server node” on page 95.

Customization Dialog walkthrough for migrating a stand-alone Application Server node

This article contains a walkthrough that describes using the new migration option within the Customization Dialog to generate the JCL jobs (CNTL/DATA data sets) for migrating your stand-alone Application Server node. Default values may be accepted for all fields and settings except those which are explicitly named below.

The steps outlined below will walk you through the process of creating the migration jobs for your stand-alone Application Server node. You must complete this walkthrough and create these jobs before you can begin the physical migration.

Customization Dialog walkthrough for stand-alone Application Server node

1. Invoke the WebSphere Application Server for Version 6.0.1 Customization Dialog:

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

   - Example *product_hlq*: WAS601.WAS
   - Example *options*: appl(az)

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

   _____

2. From the first menu page, select option **4**: Migrate V5.x Nodes to V6 Nodes.

   _____

3. On the next panel, select option **1**: Migrate a V5.x stand-alone application server node to V6.

   _____

4. Allocate partitioned datasets that you will use to store the generated migration jobs and supporting data. On the ″V5.x stand-alone Application Server Migration″ menu, select option **1**: Allocate target data sets.

   _____

5. On the ″Allocate Target Data Sets″ panel, specify your high level qualifier, and then press Enter to proceed to the next panel. Accept the defaults on the next two panels that specify the parameters for the .CNTL and .DATA data sets.

   _____

6. Back on the ″V5.x stand-alone Application Server Migration″ menu, select option **2**: Define variables. In the following panels, the migration data collection process begins.

   _____

7. On the ″Define Variables to migrate a V5.x stand-alone Application Server Node″ menu, select option **1**: System Locations (directories, HLQs, etc.).

   _____

8. The System Locations (1 of 2) panel collects information about your V6 installation libraries, and whether you intend to place the load modules in STEPLIB. Correctly specifying STEPLIB is essential to a successful migration. It is likely that your V5.x modules are currently in LPA/LNKLST, and that you will begin with your V6 libraries being defined in STEPLIB. Specify whether to put your V6 modules in STEPLIB and continue to the next panel.

   Example:

   ```
   System Locations (1 of 2)

     Specify the following V6.0.1 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.:  WAS601.WAS.SBBOLPA
SBBOLOAD:  WAS601.WAS.SBBOLOAD
SBBOLD2.:  WAS601.WAS.SBBOLD2

                              Use STEPLIB?
SCEERUN.:  CEE.SCEERUN              Y
SCEERUN2:  CEE.SCEERUN2             Y
SGSKLOAD:  SYS1.GSK.SGSKLOAD        Y
     (leave SGSKLOAD blank if all systems are at z/OS 1.6 or above)
```

9. On the ″System Locations (2 of 2)″ panel, specify your V6 WebSphere Application Server SMP/E home directory (example: `/usr/lpp/zWebSphere/V6R0`). After specifying your V6.0.1 home directory, press Enter to proceed. You will be taken back to the ″Define Variables to migrate a V5.x stand-alone Application Server Node″ menu.

10. Back on the ″Define Variables to migrate a V5.x Stand-alone Application Server node″ menu, option 1 is marked as completed. Now select option **2**: System Environment Customization.

11. The System Environment Customization panel is where you specify the configuration root HFS, which is where the configuration for the node being migrated is physically stored. You may choose to use an existing HFS if you already have an appropriate HFS on the node being migrated. If you choose to use an existing HFS, you need to ensure that the mount point you specify here is present prior to running the migration utilities (BBOWMG1B, BBOWMG2B, etc.) that are created through these dialogs. If you choose to create a new HFS on the node being migrated, the actual creation of the new HFS will not occur until you run the BBOWMBMT job during the migration process, after you complete this walkthrough (see “Migrating a base Application Server node” on page 99 for more information). Specify either an existing mount point or a new mount point and press Enter to proceed.

    Example:

```
System Environment Customization

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

 WebSphere Application Server for z/OS HFS Information

   Mount point....:  /WebSphere/V6R0
   Name...........:  OMVS.WAS.CONFIG.HFS
   Volume, or '*' for SMS.:  *
   Primary allocation in cylinders...:  250
   Secondary allocation in cylinders.:  100
```

12. Back on the ″Define Variables to migrate a V5.x Stand-alone Application Server node″ menu, options 1 and 2 are marked as completed. Now select option **3**: Server Customization.

13. On the Server Customization (1 of 2) panel, specify the source node that you are migrating under ″V5.x WebSphere Application Server home directory.″ Also, specify the home location of the profile that will contain your V6.0.1 migrated node under ″V6 WebSphere Application Server home directory.″

    On this panel, you can also choose to enable tracing on your migration utilities, which will then remain enabled throughout the entire migration process. Enabling tracing generates a lot of output, and is generally intended for use only when directed by service support.

During migration, a backup copy of the V5.x configuration is required. The default location of this backup is already specified, though you may override if needed. You may need to specify a location other than the default if the /tmp HFS does not have adequate space to store the backup configuration. If you choose to override the default location of the backup copy, IBM recommends keeping the same naming convention and just replacing the /**tmp** portion with another path, e.g., /**myTemp**/migrate/47002/base_backup. The five-digit number (47002 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/47002 (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.

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

Example:

```
Server Customization (1 of 2)

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

   V5.x WebSphere Application Server home directory:
      /WebSphere/V5R1M0
         / AppServer

   V6 WebSphere Application Server home directory:
      /WebSphere/V6R0
         / AppServer

   Migration Options

     Enable z/OS Migration Tracing:  N
     Enable WASProfile Tracing....:  N
     Enable WASPreUpgrade Tracing.:  N
     Enable WASPostUpgrade Tracing:  N

     Default Backup Directory:  /tmp/migrate/47002/base_backup
     User Specified Backup Directory:
        ==>
```

---

14. On the Server Customization (2 of 2) panel, specify the High Availability Manager Host, a new feature in V6.0.1 that has no previous value from V5.x. The value you enter here is the IP address of the LPAR you are migrating. This value must resolve to a single IP address, and you can use a dotted decimal address.

    The procedure names used to start the V6.0.1 servers are also specified on this panel; new procedure names are required for V6.0.1. You may keep the defaults, or change them if needed. The Customization Dialog creates new V6.0.1 JCL using the procedure names specified here.

    Specify values or accept defaults and press Enter to proceed.

    Example:

```
Server Customization (2 of 2)

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

   High Availability Manager Host:
     The High Availability Manager Host MUST resolve to a single
     IP address.  It can not be a multihomed host.

   Daemon Procedure name.........:  BBO6DMN

   Controller Procedure name.....:  BBO6ACR

   Servant Procedure name........:  BBO6ASR
```

15. After completing the Stand-alone Server Customization (2 of 2) panel, you will be returned to the ″Define Variables to migrate a V5.x stand-alone Application Server node″ menu. Press PF3 to return to the ″V5.x stand-alone Application Server Migration″ menu.

16. On the ″V5.x Stand-alone Application Server Migration″ menu, select option **3**: Generate customization jobs.

17. 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.

    Now you need to provide a job card customized to your environment-specific requirements.

    When you have entered your 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 V6.0.1. These jobs do not require any editing; they are to be submitted ″as-is″ during the migration process.

    Enter an appropriate job card, and press Enter to proceed.

    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
//*
//*
//*
```

18. A list of JCL jobs that have been generated for your migration is displayed. 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 BBOWMBMT successfully created.
Member BBOMBCR successfully created.
Member BBOMBCRZ successfully created.
Member BBOMBDN successfully created.
Member BBOMBDNZ successfully created.
Member BBOMBSR successfully created.
Member BBOMBSRZ successfully created.
Member BBOMBCP successfully created.
Member BBOMBINS successfully created.
```

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

---

19. You are now back on the ″V5.x stand-alone Application Server Migration″ menu. IBM recommends that you save your configuration variables for future reference by selecting option **S**: Save Customization Variables.

---

20. You have now completed the job generation process and are ready to begin the process of migration. See instructions for performing the migration by selecting option **4**: View instructions. Also, proceed to the article “Migrating a base Application Server node” for instructions on performing the migration.

Migrating a base Application Server node

This article walks you through the process of migrating a base Application Server node from V5.x to V6.0.1. Make sure that you go through the steps outlined in “Customization Dialog walkthrough for migrating a stand-alone Application Server node” on page 95 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 filesystem.

Create and mount a new V6.0.1 HFS

Before you perform the migration, V6.0.1 requires an HFS to be present for your new configuration. You can run BBOWMBMT to create and mount a new HFS, or you can mount one manually. Either way, you must have an HFS for your V6.0.1 configuration created and mounted before you proceed. This HFS is the target of the migration; your V5.x configuration HFS is the source.

BBOWMBMT creates a mount point directory, allocates the configuration’s HFS, and mounts the HFS at whatever value you specified on the field called ″Mount Point″ in the ″System Environment Customization″ panel in the Customization Dialog.

Before you proceed, ensure that you have, either manually or using BBOWMBMT, allocated, created, and mounted your HFS data sets. The mount point should be owned by the WebSphere Admin ID, and have permissions of at least 755. The new HFS structures in should be included in BPXPARM so that they will be mounted at the next IPL.

Copy your generated JCL procedures

The migration utility BBOMBCP copies the generated JCL procedures to start the servers to the specified procedure library. Your V6.0.1 configuration must make use of different JCL start procedure names; this utility will update the new V6.0.1 configuration, substituting your new JCL names in place of the names that existed in your original V5.x configuration. Submit BBOMBCP, and verify a return code of 0.

Update your RACF STARTED profiles

The STARTED profile used by controller regions is based on the procedure name and JOBNAME. Because V6.0.1 requires new start procedures, 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 V5.x controller JCL procedure name is AZACR, and you specified AZA6CR for V6.0.1, then you would need to create a STARTED profile for that new procedure name:

```
              new controller      same identity used in
                 JCL name         V5.x 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, for example, STARTED AZ*.* ... , 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.

Submit BBOWMG1B

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

Unlike in previous WebSphere Application Server for z/OS migrations, you no longer have to customize the migration utilities. This makes the migration process much more simple. BBOWMG1B, for example, is to be submitted as is.

BBOWMG1B enables all servers on the base 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. To enable PRR processing mode, follow these steps:

1. Stop the base Application Server
2. Submit the job BBOWMG1B as is, verify return code of 0
3. Restart the base Application Server, wait for it to perform PRR processing and terminate automatically

Submit BBOWMG2B

BBOWMG2B disables PRR mode and returns all servers to normal operating state. Submit this job and verify a return code of 0. You do not need to start the servers again after this job completes.

Submit BBOWMG3B

BBOWMG3B is the job that performs the physical migration of the V5.x node to V6.0.1, based on the information 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 temp directory on the HFS (this directory is `/tmp/migrate/`*`directory`*, where *directory* is the numeric value specified in the Base Server Customization panel in the Customization Dialog.

Start the base Application Server node

Use the existing commands that you would use to start a V5.x Application Server, but replace the RACF STARTED procedure name with the value you entered in the Base Server Customization Dialog (2 of 2) for Controller Procedure name. This command starts the V6.0.1 base Application Server. Wait until the server is finished initializing before proceeding.

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

```
BBOO0019I INITIALIZATION COMPLETE FOR WEBSPHERE FOR z/OS CONTROL PROCESS BBOS001
```

At this point, your migration to V6.0.1 is complete.

Perform post-migration tasks

After you have verified a successful migration to V6.0.1, and are successfully running a migrated configuration, you should delete:

- everything in the source configuration's HFS
- everything in the target configuration's /tmp/migrate/*nnnnn*/ directory
- bbomigrt2.sh

Checklist of migration activities for base Application Server node

This checklist is intended to provide a quick reference.

More details can be found under "Preparing to migrate a base Application Server node to V6.0.1" on page 92.

Key similarities with an ND configuration:

- You must still create ″skeleton″ CNTL/DATA data sets through the Customization Dialog.
- You must still allocate and mount a new HFS to act as the target for the new V6.0.1 configuration, either manually or using BBOWMBMT.
- You must still run BBOWMG3B to perform the migration.

Key differences from an ND configuration:

- You must clear 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.

General preparation work

Table 10.

Check off	Item
	Insure minimum level of Version 6.0.1 maintenance. See "Overview of the V6.0.1 migration process" on page 89.
	Inventory your existing V5.x configuration and capture key information (see "Preparing to migrate a base Application Server node to V6.0.1" on page 92).
	Create ″skeleton″ CNTL/DATA data sets for each node to be migrated (see "Customization Dialog walkthrough for migrating a stand-alone Application Server node" on page 95).
	Create V6.0.1 configuration mount point, allocate HFS and mount either manually or using BBOWMBMT.
	Run BBOMBCP to modify JCL start procedure names.

Clear transaction logs for Application Server node

Check off	Item
	Stop server in the Base Application server node.
	Run BBOWMG1B. Check for success.
	Restart Application Server to clear transaction logs. It will start and then automatically stop.
	Run BBOWMG2B to disable PRR mode.

Migrate Application Server node

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

Network Deployment migrations

The migration documentation for a deployment manager includes planning information, a Customization Dialog walkthrough, a detailed V6.0.1 migration explanation, and a very high-level migration checklist. Select the appropriate link for information about how to migrate to a V6.0.1 deployment manager:

- "Preparing to migrate a Network Deployment configuration to V6.0.1"
- "Customization Dialog walkthrough for deployment manager" on page 104
- "Migrating a deployment manager" on page 109
- "Checklist of migration activities for a Network Deployment configuration" on page 111

You can use the migration utilities to migrate a Version 5.x deployment manager to WebSphere Application Server for z/OS Version 6.0.1.

Return to Chapter 4, "Migrating product configurations," on page 89 to continue.

Preparing to migrate a Network Deployment configuration to V6.0.1

Read the following information carefully to be sure that you are ready to migrate to V6.0.1.

WebSphere Application Server for z/OS Version 6.0.1 supplies migration utilities in the data set that you specify during the ″Allocate target data sets″ step in the Customization Dialog (see "Overview of the V6.0.1 migration process" on page 89 for a description of each utility):

- BBOWMDMT
- BBOWMG3D
- BBOMDCP

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

Inventory your existing configuration

WebSphere Application Server for z/OS V5.x configuration general information:

Cell short name:	

Cell long name:	
Sysplex name:	
HFS strategy:	__ All nodes share one HFS __ Unique HFS per node
JCL start procedure strategy:	__ Use common JCL __ Unique JCL per server
Location of original V5.x configuration datasets:	
WebSphere Application Server V5.x home directory (configuration HFS):	

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

WebSphere Application Server for z/OS V5.x Deployment Manager node:

Configuration mount point:	
Configuration HFS data set:	
Home directory:	default:/DeploymentManager your value: _____
WebSphere Application Server SMP/E home:	default: /usr/lpp/zWebSphere/V601M2 your value: _____
Controller JCL start proc:	
Servant JCL start proc:	
Daemon JCL start proc:	

WebSphere Application Server for z/OS V6.0.1 installation information:

SMP/E data set HLQ:	
SMP/E HFS Mount Point:	
WebSphere Application Server V6.0.1 product directory (SMP/E target for V6.0.1):	
WebSphere Application Server V6.0.1 in LPA/LNKLST?:	__ No __ Yes

Plan the node-by-node migration strategy

"Overview of the V6.0.1 migration process" on page 89 explains the order of migration for a Network Deployment configuration.

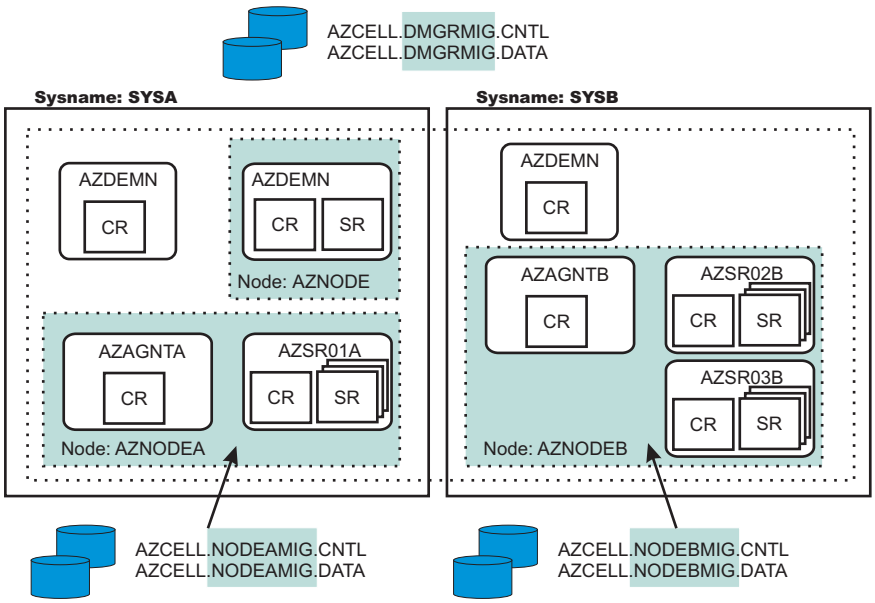
Take a moment to map out *your* migration sequence:

	Node Short Name	***System Name***
First node		
Second node		
Third node		

Fourth node		
Fifth node		
Sixth node		

Produce skeleton 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 Dialogs for V6.0.1 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, on most of the panels it will be necessary to supply information related to your configuration. 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 some key information about your new V6.0.1 configuration (for example, mount points and paths), 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 (for example: settings and applications) from your existing V5.x configuration HFS.

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 walkthrough for deployment manager.”

Customization Dialog walkthrough for deployment manager

This article contains a walkthrough that describes using the Customization Dialog to generate the JCL jobs (CNTL/DATA data sets) for migrating your deployment manager. Default values may be accepted for all fields and settings except those which are explicitly named below.

The steps outlined below will walk you through the process of creating the migration jobs for your deployment manager. You must complete this walkthrough and create these jobs before you can begin the physical migration.

Customization Dialog walkthrough for a deployment manager

1. Invoke the WebSphere Application Server for Version 6.0.1 Customization Dialog:

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

   - Example *product_hlq*: WAS601.WAS
   - Example *options*: appl(az)

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

   _____

2. From the first menu page, select option **4**: Migrate V5.x Nodes to V6 Nodes.

   _____

3. On the next panel, select option **2**: Migrate a V5.x deployment manager to V6.

   _____

4. The next step is to allocate partitioned datasets that you will use to store the generated migration jobs and supporting data. On the ″V5.x Deployment Manager Migration″ menu, select option **1**: Allocate target data sets.

   _____

5. On the ″Allocate Target Data Sets″ panel, specify your high level qualifier, and then press Enter to proceed to the next panel. Accept the defaults on the next two panels that specify the parameters for the .CNTL and .DATA data sets.

   _____

6. Back on the ″V5.x Deployment Manager Migration″ menu, select option **2**: Define variables. In the following panels, the migration data collection process begins.

   _____

7. On the ″Define Variables to migrate a V5.x Deployment Manager Node″ menu, select option **1**: System Locations (directories, HLQs, etc.).

   _____

8. The System Locations (1 of 2) panel collects information about your V6 installation libraries, and whether you intend to place the load modules in STEPLIB. Correctly specifying STEPLIB is essential to a successful migration. It is likely that your V5.x modules are currently in LPA/LNKLST, and that you will begin with your V6 libraries being defined in STEPLIB. Specify whether to put your V6 modules in STEPLIB and press Enter to continue to the next panel.

   Example:

```
System Locations (1 of 2)

  Specify the following V6.0.1 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.:  WAS601.WAS.SBBOLPA
SBBOLOAD:  WAS601.WAS.SBBOLOAD
SBBOLD2.:  WAS601.WAS.SBBOLD2

                              Use STEPLIB?
SCEERUN.:  CEE.SCEERUN              Y
```

```
SCEERUN2:  CEE.SCEERUN2             Y
SGSKLOAD:  SYS1.GSK.SGSKLOAD        Y
     (leave SGSKLOAD blank if all systems are at z/OS 1.6 or above)
```

9. On the ″System Locations (2 of 2)″ panel, specify your V6 WebSphere Application Server SMP/E home directory (example: `/usr/lpp/zWebSphere/V6R0`). After specifying your V6.0.1 home directory, press Enter to proceed. You will be taken back to the ″Define Variables to migrate a V5.x stand-alone Application Server Node″ menu.

10. Back on the ″Define Variables to migrate a V5.x Stand-alone Application Server node″ menu, option 1 is marked as completed. Now select option **2**: System Environment Customization.

11. The System Environment Customization panel is where you specify the configuration root HFS, which is where the configuration for the node being migrated is physically stored. You may choose to use an existing HFS if you already have an appropriate HFS on the node being migrated. If you choose to use an existing HFS, you need to ensure that the mount point you specify here is present prior to running the migration utilities (BBOWMDMT, BBOMDCP, etc.) that are created through these dialogs. If you choose to create a new HFS on the node being migrated, the actual creation of the new HFS will not occur until you run the BBOWMBMT job during the migration process, after you complete this walkthrough (see “Migrating a base Application Server node” on page 99 for more information). Specify either an existing mount point or a new mount point and press Enter to proceed.

    Example:

```
System Environment Customization

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

 WebSphere Application Server for z/OS HFS Information

   Mount point....:  /WebSphere/V6R0
   Name...........:  OMVS.WAS.CONFIG.HFS
   Volume, or '*' for SMS.:  *
   Primary allocation in cylinders...:  250
   Secondary allocation in cylinders.:  100
```

12. Back on the ″Define Variables to migrate a V5.x Deployment Manager Node″ menu, options 1 and 2 are marked as completed. Now select option **3**: Server Customization.

13. On the Server Customization (1 of 2) panel, specify the source node that you are migrating under ″V5.x WebSphere Application Server home directory.″ Also, specify the home location of the profile that will contain your V6.0.1 migrated node under ″V6 WebSphere Application Server home directory.″

    On this panel, you can also choose to enable tracing on your migration utilities, which will then remain enabled throughout the entire migration process.

    During migration, a backup copy of the V5.x configuration is required. The default location of this backup is already specified, though you may override if needed. You may need to specify a location other than the default if the /tmp HFS does not have adequate space to store the backup configuration. If you choose to override the default location of the backup copy, IBM recommends keeping the same naming convention and just replacing the **/tmp** portion with another path, e.g., /**myTemp**/migrate/61036/dmgr_backup. The five-digit number (61036 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/61036 (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.

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

Example:

```
Server Customization (1 of 2)

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

   V5.x WebSphere Application Server home directory:
      /WebSphere/V5R1M0
         / DeploymentManager

   V6 WebSphere Application Server home directory:
      /WebSphere/V6R0
         / DeploymentManager

   Migration Options

     Enable z/OS Migration Tracing:  N
     Enable WASProfile Tracing....:  N
     Enable WASPreUpgrade Tracing.:  N
     Enable WASPostUpgrade Tracing:  N

     Default Backup Directory:  /tmp/migrate/61036/dmgr_backup
     User Specified Backup Directory:
        ==>
```

---

14. On the Server Customization (2 of 2) panel, specify the High Availability Manager Host, a new feature in V6.0.1 that has no previous value from V5.x.

    The procedure names used to start the V6.0.1 servers are also specified on this panel; new procedure names are required for V6.0.1. You may keep the defaults, or change them if needed. The Customization Dialog creates new V6.0.1 JCL using the procedure names specified here.

    Specify values or accept defaults and press Enter to proceed.

    Example:

```
Server Customization (2 of 2)

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


   High Availability Manager Host:
     The High Availability Manager Host MUST resolve to a single
     IP address.  It can not be a multihomed host.

   Daemon Procedure name........:  BBO6DMN

   Controller Procedure name....:  BBO6DCR

   Servant Procedure name.......:  BBO6DSR
```

---

15. After completing the Stand-alone Server Customization (2 of 2) panel, you will be returned to the ″Define Variables to migrate a V5.x stand-alone Application Server node″ menu. Press PF3 to return to the ″V5.x stand-alone Application Server Migration″ menu.

---

16. On the ″V5.x Deployment Manager Migration″ menu, select option **3**: Generate customization jobs.

---

17. 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.

    Now you need to provide a job card customized to your environment-specific requirements.

When you have entered your 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 V6.0.1. These jobs do not require any editing; they are to be submitted ″as-is″ during the migration process.

Enter an appropriate job card, and press Enter to proceed.

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
//*
//*
//*
```

---

18. A list of JCL jobs that have been generated for your migration is displayed. When processing has completed, press Enter to proceed.

    Example:

    ```
    Processing for data set 'hlq.CNTL' ...
    Member BBOWMG3D successfully created.
    Member BBOWMDMT 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.
    ***
    ```

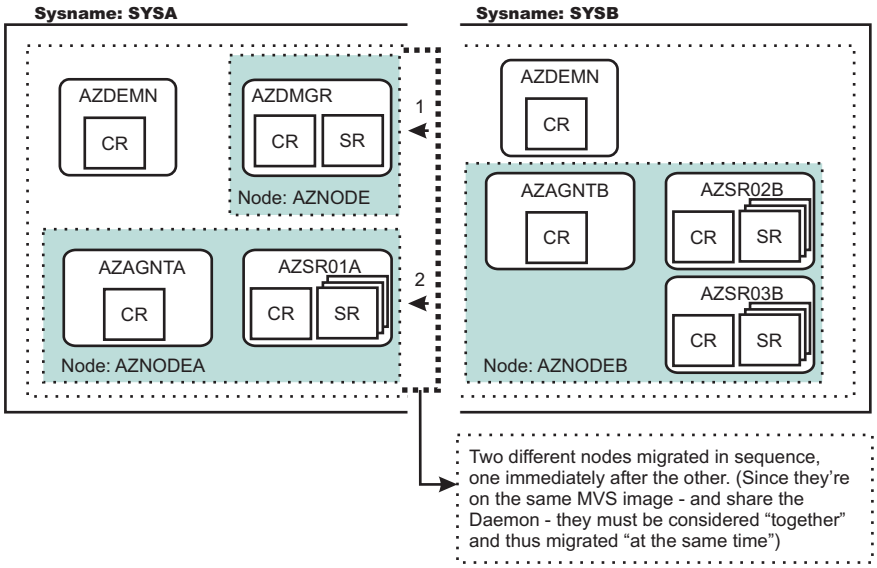
---

19. You are now back on the ″V5.x Deployment Manager Migration″ menu. IBM recommends that you save your configuration variables for future reference by selecting option **S**: Save Customization Variables.
20. You have now completed the job generation process and are ready to begin the process of migration. See instructions for performing the migration by selecting option **4**: View instructions. Also, proceed to the article “Migrating a base Application Server node” on page 99 for instructions on performing the migration.

Migrating a deployment manager

The Deployment Manager node should always be migrated first. Even though each node is migrated separately, whenever you have two more nodes from the same cell on the same MVS image, you should migrate them 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 filesystem.



Create and mount a new V6.0.1 HFS

Before you perform the migration, V6.0.1 requires an HFS to be present for your new configuration. You can run BBOWMDMT to create and mount a new HFS, or you can mount one manually. Either way, you must have an HFS for your V6.0.1 configuration created and mounted before you proceed. This HFS is the target of the migration; your V5.x configuration HFS is the source.

BBOWMDMT creates a mount point directory, allocates the configuration's HFS, and mounts the HFS at whatever value you specified on the field called ″Mount Point″ in the ″System Environment Customization″ panel in the Customization Dialog.

Before you proceed, ensure that you have, either manually or using BBOWMDMT, allocated, created, and mounted your HFS data sets. The mount point should be owned by the WebSphere Admin ID, and have permissions of at least 755. The new HFS structures in should be included in BPXPARM so that they will be mounted at the next IPL.

Copy your generated JCL procedures

The migration utility BBOMDCP copies the generated JCL procedures to start the servers to the specified procedure library. Your V6.0.1 configuration must make use of different JCL start procedure names; this utility will update the new V6.0.1 configuration, substituting your new JCL names in place of the names that existed in your original V5.x configuration. Submit BBOMDCP, and verify a return code of 0.

Update your RACF STARTED profiles

The STARTED profile used by controller regions is based on the procedure name and JOBNAME. Because V6.0.1 requires new start procedures, 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 V5.x deployment manager

controller JCL procedure name is AZDCR, and you specified AZ1DCR for V6.0.1, then you would need to create a STARTED profile for that new procedure name:

```
              new controller      same identity used in
                 JCL name         V5.x configuration
                    |                    |
 RDEFINE STARTED AZ1DCR.* STDATA(USER(AZDCRU) 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, for example, STARTED AZ*.* ... , 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.

Submit BBOWMG3D

A deployment manager migration does not require bringing the node into and out of PRR mode as base Application Server and federated node migrations do. Hence, there are two less jobs to submit for a deployment manager migration, and, at this point, you are ready to perform the physical migration.

BBOWMG3D is the job that performs the physical migration of the V5.x deployment manager to V6.0.1, 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 HFS (this directory is `/tmp/migrate/`*`directory`*, where *directory* is the numeric value specified in the Deployment Manager Customization panel in the Customization Dialog.

Update deployment manager's servant keyring

If you use separate user IDs to run your deployment manager's controller and servant regions, you must add a WASKeyring to the deployment manager's servant user ID, then connect the WebSphere CA certificate to the WASKeyring:

```
RACDCERT ADDRING(WASKeyring) ID(dmgr_servant_user_ID)
RACDCERT ID(dmgr_servant_user_ID) CONNECT(RING(WASKeyring) LABEL('WebSphereCA') CERTAUTH)
```

Start the deployment manager

Use the existing commands that you would use to start a V5.x Application Server, but replace the RACF STARTED procedure name with the value you entered in the Deployment Managre Customization Dialog (2 of 2) for Controller Procedure name. This command starts the V6.0.1 deployment manager. Wait until the server is finished initializing before proceeding.

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

```
BBOO0019I INITIALIZATION COMPLETE FOR WEBSPHERE FOR z/OS CONTROL PROCESS BBOS001
```

At this point, your migration to V6.0.1 is complete.

Perform post-migration tasks

After you have verified a successful migration to V6.0.1, and are successfully running a migrated configuration, you should delete:

- everything in the source configuration's HFS
- everything in the target configuration's /tmp/migrate/*nnnnn*/ directory

- bbomigrt2.sh

Checklist of migration activities for a Network Deployment configuration

This checklist is intended to provide a quick reference.

More details can be found here: “Preparing to migrate a Network Deployment configuration to V6.0.1” on page 102.

General preparation work

Table 11.

Check off	Item
	Inventory your existing V5.x configuration and capture key information.
	Map strategy for the sequence of nodes to be migrated.
	Create ″skeleton″ CNTL/DATA data sets for each node to be migrated through Customization Dialog (see “Customization Dialog walkthrough for deployment manager” on page 104).
	Create V6.0.1 configuration mount point, allocate HFS and mount either manually or using BBOWMDMT.
	Customize and run BBOMDCP to modify JCL start procedure names used.

Migrate deployment manager node

Check off	Item
	Stop deployment manager plus associated Daemon server.
	Run BBOWMG3D and check output for success.
	Customize and run BBOWBPR to modify JCL start procedure names used.
	Start V6.0.1 copy of Deployment Manager.

Managed (federated) node migrations

The migration documentation for a federated node includes a Customization Dialog walkthrough, a detailed V6.0.1 migration explanation, and an explanation on migrating a federated node another MVS image. Select the appropriate link for information about how to migrate to a V6.0.1 federated Application Server node:

- “Customization Dialog walkthrough for migrating a managed Application Server node” on page 112
- “Migrating a federated Application Server node” on page 116
- “Migrating an Application Server node on another MVS image” on page 118

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

Return to Chapter 4, “Migrating product configurations,” on page 89 to continue.

Customization Dialog walkthrough for migrating a managed Application Server node

This article contains a walkthrough that describes using the new migration option within the Customization Dialog to generate the JCL jobs (CNTL/DATA data sets) for migrating your managed (federated) Application Server node. Default values may be accepted for all fields and settings except those which are explicitly named below. Before migrating a managed node, you **MUST** migrate its deployment manager first. If you have not migrated the deployment manager, see “Network Deployment migrations” on page 102, then return to this article after the deployment manager is migrated.

The steps outlined below will walk you through the process of creating the migration jobs for your managed Application Server node. You must complete this walkthrough and create these jobs before you can begin the physical migration.

Customization Dialog walkthrough for managed Application Server node

1. Invoke the WebSphere Application Server for Version 6.0.1 Customization Dialog:

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

   - Example *product_hlq*: WAS601.WAS
   - Example *options*: appl(az)

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

   _____

2. From the first menu page, select option **4**: Migrate V5.x Nodes to V6 Nodes.

   _____

3. On the next panel, choose option **3**: Migrate a V5.x managed application server node to V6.

   _____

4. The next step is to allocate partitioned datasets that you will use to store the generated migration jobs and supporting data. On the ″V5.x Federated Node Migration″ menu, choose option **1**: Allocate target data sets.

   _____

5. On the ″Allocate Target Data Sets″ panel, specify your high level qualifier, and then press Enter to proceed to the next panel. Accept the defaults on the next two panels that specify the parameters for the .CNTL and .DATA data sets.

   _____

6. Back on the ″V5.x Federated Node Migration″ menu, choose option **2**: Define variables. In the following panels, the migration data collection process begins.

   _____

7. On the ″Define Variables to migrate a V5.x Federated Node″ menu, choose option **1**: System Locations (directories, HLQs, etc.).

   _____

8. The System Locations (1 of 2) panel collects information about your V6 installation libraries, and whether you intend to place the load modules in STEPLIB. Correctly specifying STEPLIB is essential to a successful migration. It is likely that your V5.x modules are currently in LPA/LNKLST, and that you will begin with your V6 libraries being defined in STEPLIB. Specify whether to put your V6 modules in STEPLIB and continue to the next panel.

   Example:

   ```
   System Locations (1 of 2)

     Specify the following V6.0.1 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.:  WAS601.WAS.SBBOLPA
SBBOLOAD:  WAS601.WAS.SBBOLOAD
SBBOLD2.:  WAS601.WAS.SBBOLD2

                              Use STEPLIB?
SCEERUN.:  CEE.SCEERUN              Y
SCEERUN2:  CEE.SCEERUN2             Y
SGSKLOAD:  SYS1.GSK.SGSKLOAD        Y
     (leave SGSKLOAD blank if all systems are at z/OS 1.6 or above)
```

9. On the ″System Locations (2 of 2)″ panel, specify your V6 WebSphere Application Server SMP/E home directory (example: `/usr/lpp/zWebSphere/V6R0`). After specifying your V6.0.1 home directory, press Enter to proceed. You will be taken back to the ″Define Variables to migrate a V5.x managed Application Server Node″ menu.

10. Back on the ″Define Variables to migrate a V5.x Federated Node″ menu, option 1 is marked as completed. Now choose option **2**: System Environment Customization.

11. The System Environment Customization panel is where you specify the configuration root HFS, which is where the configuration for the node being migrated is physically stored. You may choose to use an existing HFS if you already have an appropriate HFS on the node being migrated. If you choose to use an existing HFS, you need to ensure that the mount point you specify here is present prior to running the migration utilities (BBOWMG1F, BBOWMG2F, etc.) that are created through these dialogs. If you choose to create a new HFS on the node being migrated, the actual creation of the new HFS will not occur until you run the BBOWMMMT job during the migration process, after you complete this walkthrough (see “Migrating a base Application Server node” on page 99 for more information). Specify either an existing mount point or a new mount point and press Enter to proceed.

    Example:

```
System Environment Customization

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

 WebSphere Application Server for z/OS HFS Information

   Mount point....:  /WebSphere/V6R0
   Name...........:  OMVS.WAS.CONFIG.HFS
   Volume, or '*' for SMS.:  *
   Primary allocation in cylinders...:  250
   Secondary allocation in cylinders.:  100
```

12. Back on the ″Define Variables to migrate a V5.x managed Application Server node″ menu, options 1 and 2 are marked as completed. Now select option **3**: Server Customization.

13. On the Server Customization (1 of 2) panel, specify the source node that you are migrating under ″V5.x WebSphere Application Server home directory.″

    Also, specify the home location of the profile that will contain your V6.0.1 migrated node under ″V6 WebSphere Application Server home directory.″

On this panel, you can also choose to enable tracing on your migration utilities, which will then remain enabled throughout the entire migration process. Enabling tracing generates a lot of output, and is generally intended for use only when directed by service support.

During migration, a backup copy of the V5.x configuration is required. The default location of this backup is already specified, though you may override if needed. You may need to specify a location other than the default if the /tmp HFS does not have adequate space to store the backup configuration. If you choose to override the default location of the backup copy, IBM recommends keeping the same naming convention and just replacing the /**tmp** portion with another path, e.g., /**myTemp**migrate/57638/fed_backup. The five-digit number (57638 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/57638 (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.

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

Example:

```
Server Customization (1 of 2)

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

   V5.x WebSphere Application Server home directory:
      /WebSphere/V5R1M0
         / AppServer

   V6 WebSphere Application Server home directory:
      /WebSphere/V6R0
         / AppServer

   Migration Options

     Enable z/OS Migration Tracing:  N
     Enable WASProfile Tracing....:  N
     Enable WASPreUpgrade Tracing.:  N
     Enable WASPostUpgrade Tracing:  N

     Default Backup Directory:  /tmp/migrate/57638/fed_backup
     User Specified Backup Directory:
        ==>
```

14. On the Server Customization (2 of 2) panel, specify the High Availability Manager Host, a new feature in V6.0.1 that has no previous value from V5.x. The value you enter here is the IP address of the LPAR you are migrating. This value must resolve to a single IP address, and you can use a dotted decimal address.

    The procedure names used to start the V6.0.1 servers are also specified on this panel; new procedure names are required for V6.0.1. You may keep the defaults, or change them if needed. The Customization Dialog creates new V6.0.1 JCL using the procedure names specified here.

    Because some of the migration jobs require running under a WebSphere Administrator ID, you need to supply a valid administrator ID and password here as well.

    Specify values or accept defaults and press Enter to proceed.

    Example:

```
Server Customization (2 of 2)

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


   High Availability manager Host:
```

```
  The High Availability Manager Host MUST resolve to a single
  IP address.  It can not be a multihomed host.

Daemon Procedure name.........:  BBO6DMN

Controller Procedure name.....:  BBO6ACR

Servant Procedure name........:  BBO6ASR

Some migration tasks require running under the
WebSphere Administrators account:

  WebSphere Administrator User ID.:  XXXXXXX
  WebSphere Administrator Password:  XXXXXXXX
```

---

15. After completing the managed Server Customization (2 of 2) panel, you will be returned to the ″Define Variables to migrate a V5.x Federated Node″ menu. Press PF3 to return to the ″V5.x Federated Node Migration″ menu.

---

16. On the ″V5.x Federated Node Migration″ menu, choose option **3**: Generate customization jobs.

---

17. 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.

    Now you need to provide a job card customized to your environment-specific requirements.

    When you have entered your 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 V6.0.1. These jobs do not require any editing; they are to be submitted ″as-is″ during the migration process.

    Enter an appropriate job card, and press Enter to proceed.

    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
//*
//*
//*
```

---

18. A list of JCL jobs that have been generated for your migration is displayed. 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 BBOWMMMT 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.
***
```

---

19. You will be returned back to the ″V5.x Federated Node Migration″ menu. IBM recommends that you now save your configuration variables for future reference by selecting option **S**: Save Customization Variables.

---

20. You have now completed the job generation process and are ready to begin the process of migration. See instructions for performing the migration by selecting option **4**: View instructions. Also, proceed to the article “Migrating a federated Application Server node” for instructions on performing the migration.

Migrating a federated Application Server node

With the Deployment Manager migrated and restarted, you are ready to migrate a federated Application Server node as well.

Note: At this point, ensure that the Application Servers and the node agent are stopped.

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 filesystem.

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 Version 6.0.1 deployment manager. This validates that the deployment manager is running.

Check Off	Item
	Ensure that the Version 6.0.1 copy of the code is running. Under ″About your WebSphere Application Server″, the build number should begin with `W601`.

Create and mount a new V6.0.1 HFS

Before you perform the migration, V6.0.1 requires an HFS to be present for your new configuration. You can run BBOWMMMT to create and mount a new HFS, or you can mount one manually. Either way, you must have an HFS for your V6.0.1 configuration created and mounted before you proceed. This HFS is the target of the migration; your V5.x configuration HFS is the source.

BBOWMMMT creates a mount point directory, allocates the configuration's HFS, and mounts the HFS at whatever value you specified on the field called ″Mount Point″ in the ″System Environment Customization″ panel in the Customization Dialog.

Before you proceed, ensure that you have, either manually or using BBOWMMMT, allocated, created, and mounted your HFS data sets. The mount point should be owned by the WebSphere Admin ID, and have permissions of at least 755. The new HFS structures in should be included in BPXPARM so that they will be mounted at the next IPL.

Copy your generated JCL procedures

The migration utility BBOMMCP copies the generated JCL procedures to start the servers to the specified procedure library. Your V6.0.1 configuration must make use of different JCL start procedure names; this utility will update the new V6.0.1 configuration, substituting your new JCL names in place of the names that existed in your original V5.x configuration. Submit BBOMMCP, and verify a return code of 0.

Update your RACF STARTED profiles

The STARTED profile used by controller regions is based on the procedure name and JOBNAME. Because V6.0.1 requires new start procedures, 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 V5.x controller JCL procedure name is AZACR, and you specified AZ6ACR for V6.0.1, then you would need to create a STARTED profile for that new procedure name:

```
              new controller      same identity used in
                 JCL name         V5.x 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, for example, STARTED AZ*.* ... , 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.

Submit BBOWMG1F

Note: If you are not using XA connectors, submitting BBOWMG1F and BBOWMG2F is optional. However, IBM recommends that you submit both jobs to ensure that your transaction logs are clear.

Unlike in previous WebSphere Application Server for z/OS migrations, you no longer have to customize the migration utilities. This makes the migration process much more simple. BBOWMG1F, for example, is to be submitted as is.

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. To enable PRR processing mode, follow these steps:

1. Stop the federated Application Server.
2. Submit the job BBOWMG1F as is, verify return code of 0.
3. Restart the federated Application Server, wait for it to perform PRR processing and terminate automatically.

Submit BBOWMG2F

BBOWMG2F disables PRR mode and returns all servers to normal operating state. Submit this job and verify a return code of 0. You do not need to start the servers again after this job completes.

Submit BBOWMG3F

BBOWMG3F is the job that performs the physical migration of the V5.x node to V6.0.1, 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 migration temp directory on the HFS (this directory is `/tmp/migrate/directory`, where *directory* is the numeric value specified in the Federated Node Customization panel in the Customization Dialog.

Start the managed Application Server node

Use the existing commands that you would use to start a V5.x Application Server, but replace the RACF STARTED procedure name with the value you entered in the Federated Node Customization Dialog (2 of 2) for Controller Procedure name. This command starts the V6.0.1 federated Application Server. Wait until the server is finished initializing before proceeding.

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

```
BBOO0019I INITIALIZATION COMPLETE FOR WEBSPHERE FOR z/OS CONTROL PROCESS BBOS001
```

At this point, your migration to V6.0.1 is complete.

Perform post-migration tasks

After you have verified a successful migration to V6.0.1, and are successfully running a migrated configuration, you should delete:

- everything in the target configuration's HFS
- everything in the target configuration's /tmp/migrate/*nnnnn*/ directory
- bbomigrt2.sh

Migrating an Application Server node on another MVS image

The process is nearly identical to the process outlined in "Migrating a federated Application Server node" on page 116. Follow the procedure outlined in that article, keeping aware of the following points:

- Ensure that the V6.0.1 deployment manager is up and running.
- Run the jobs on the same system on which the node itself resides.
- You must go through the migration section of the Customization Dialogs to create customized migration jobs, just as you would on the same MVS image.

Migration tools

This topic introduces the migration tools that WebSphere Application Server provides. All of the migration tools are in the `install_root/bin` directory after installation. It is important to use the migration tools for the version of Application Server 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.

clientUpgrade.sh (and clientUpgrade.bat)
Upgrades the client application to a new release level.

convertScriptCompatibility.sh (and convertScriptCompatibility.bat)
Used by administrators to convert their configuration from a mode that supports backward compatibility of V5.x administration scripts to a mode that is fully V6.0.

The clientUpgrade command

Use the **clientUpgrade** command to migrate V5.x client resources to V6.0.1 level resources. In the process of migrating these resources, the `client-resources.xmi` file located in the client jars will be migrated to the latest level. A backup of the client-resources.xmi file will also be located in the client jar. If this command is not executed against the client EAR files before they are installed on V6.0.1, the client EARs will not operate or install correctly.

The command file is located in the `bin` subdirectory of the *WAS_install_root*, or the *ND_install_root* directory. By default, the *WAS_install_root* for WebSphere Application Server for z/OS is `/usr/WebSphere/AppServer`.

By default, the *ND_install_root* for WebSphere Application Server for z/OS is `/usr/WebSphere/DeploymentManager`.

Command syntax:

```
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
```

The convertScriptCompatibility command

The **convertScriptCompatibility** command is a migration tool used by administrators for converting a configuration from a mode that supports backward compatibility of V5.x administration scripts to a mode that is fully V6.0. This command converts WebSphere Common Configuration Model (WCCM) objects of type processDef to use processDefs as defined in the 6.0 `server.xml` model. There can be only one occurrence of a processDefs object in a server configuration. If an existing processDefs object is found when performing this conversion, it is used and updated; otherwise a new object is created. The **convertScriptCompatibility** command also maps existing transport entries to channel support. This affects server.xml and serverindex.xml files. The values of the transport settings are used to create new channel entries.

convertScriptCompatibility.bat command syntax for Windows platforms

The command syntax is as follows:

```
convertScriptCompatibility  [-help]
                                [-backupConfig true | false]
                                [-profileName profile_name]
                                [-nodeName node_name [-serverName server_name]]
                                [-traceString trace_spec [-traceFile file_name]]
```

Parameters

Supported arguments include the following:

-help
Displays help for this command

-backupConfig true | false
An optional parameter used to back up the existing configuration of the current instance. The default is `true`, to use the **backupConfig** command to save a copy of the current configuration into the *profile_name*`/temp` directory. Use the **restoreConfig** command to restore that configuration as required.

-profileName *profile_name*
An optional parameter used to specify a profile configuration in the V6 environment. If not specified, the default profile will be used. If the default has not been set or cannot be found, the system will return an error.

-nodeName *node_name* **-serverName** *server_name*
Optional parameters used to specify a node name and a server name for the program to update. If neither is specified, all nodes and servers in the configuration are converted. When you use -serverName in conjunction with -nodeName, all processing will be limited to the specified *node_name*.

-traceString *trace_spec* **-traceFile** *file_name*
Optional parameters to gather trace information for IBM Service personnel. Specify a trace specification of `"*=all=enabled"` (with quotation marks) to gather all trace information.

Rolling back your environment to V5.x

This task describes rolling back a WebSphere Application Server Version 6.0.1 environment to Version 5.x.

1. To roll back a cell, from the Version 5.x directory of each node, run the following command:
   ```
   ./wsadmin.sh -f dmDisablementReversal.jacl -conntype NONE
   ```
2. Run this command from each deployment manager or node that you need to roll back from Version 6.0.1.

Chapter 5. Coexisting

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

Coexisting, as it applies to WebSphere Application Server products, is the ability of multiple installations of WebSphere Application Server to run on the same system at the same time. Multiple installations include multiple versions and multiple instances of one version. Coexistence also implies various combinations of Web server interaction.

Read about “Coexistence support.”

This topic discusses which coexistence scenarios are supported.

Coexistence support

Coexistence, as it applies to WebSphere Application Server for z/OS products, is 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; i.e., 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 following points:

General coexistence:

- Multiple WebSphere Application Server for z/OS cells can coexist in the same sysplex.
- Multiple WebSphere Application Server for z/OS nodes can coexist on the same LPAR.
- WebSphere Application Server for z/OS V5.0.x, V5.1, and V6.0.1 nodes can all coexist in the same cell.
- No two cells can have the same cell short name.
- Separate cells need separate HFS mount points and JCL procedures.
- If your V5.x load modules reside in LPA/LNKLST, you will need to place the V6.0.1 modules in STEPLIB in order to coexist.
- You need to determine port conflicts that might occur when earlier versions coexist with Version 6.0.1. See Planning a TCP/IP port convention in the *Installing your application serving environment* PDF book for default port information.

V5.0.x and V5.1 coexistence:

- WebSphere Application Server for z/OS V5.0.x and V5.1 nodes can coexist in the same cell if and only if they are on different LPARs.
- WebSphere Application Server for z/OS V5.0.x and V5.1 nodes can coexist on the same LPAR if and only if they are in different cells.

V5.0.x and V6.0.1 coexistence:

- WebSphere Application Server for z/OS V5.0.x and V6.0.1 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 V5.0.x nodes to coexist with V6.0.1 in the same cell
- WebSphere Application Server for z/OS V5.0.x and V6.0.1 nodes can coexist on the same LPAR if and only if they are in different cells.

V5.1 and V6.0.1 coexistence:

- WebSphere Application Server for z/OS V5.1 and V6.0.1 nodes can coexist on the same LPAR in the same cell or in different cells.

Chapter 6. Interoperating

WebSphere Application Server Version 6.0.1 is generally interoperable with WebSphere Application Server Versions 5.0.x and 5.1.x. IBM recommends that you apply the latest fix level to support interoperability. Currently, there are no interim fixes necessary to support interoperability. However, in order to interoperate between WebSphere Application Server for z/OS and a WebSphere Application Server product on a non-z/OS machine, you take the following steps:

1. On the non-z/OS machine, the property `-Dcom.ibm.CORBA.ORBCharEncoding=ASCII` should be set under Server1 ->process definitions-> jvm prop->generic JVM arguments.
2. On the non-z/OS machine, the property `com.ibm.CORBA.ORBWCharDefault = UCS2` should be defined under Server1->orb services->custom property.
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.

Chapter 7. Configuring ports

This topic discusses configuring ports, particularly in coexistence scenarios.

1. Review the port number settings, especially when you are planning to coexist.

   See Planning a TCP/IP port convention in the *Installing your application serving environment* PDF book for reference information for identifying port numbers used by present and past product versions.

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.

   After installation, edit the *profiles_install_root*/*profile_name*/config/cells/*cell_name*/nodes/*node_name*/serverindex.xml file to change the port settings, or use scripting to change the values.

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/).