BEA WebLogic® Integration

Upgrade Guide

Version 10.2
Document Revised: March 2008

Contents

1. Overview

2. The Upgrade Process

5. Other Component Changes

CHAPTER 1

Overview

This section includes the following topics:

Scope of this Document

This document describes the procedures required to upgrade your application environment from BEA WebLogic® Integration 8.1.x and 8.5 x and 9.x to BEA WebLogic® Integration 10.2. An application environment includes applications, the WebLogic domains in which they are deployed, and any application data associated with the domain, and may include external resources, such as database servers, firewalls, load balancers, and LDAP servers.

The WLI Environment

The WLI environment consists of the:

- WLI domain: Is a WebLogic domain that consists of a group of one or more WebLogic servers, known as Managed Servers. Managed servers are administered by a single Administration Server. The Administration Server hosts the Administration Console. WLI

applications can run on both Administration and Managed servers. A domain provides a set of configurable attributes that can be viewed and modified using the Administration Console. These attributes are stored in the config.xml file available at the location: `BEA_HOME`\user project\`domain_name`\. You can configure some attributes related to security, clustering, transaction, and logging.

- WLI applications: These are Business Processes (Java Process Definitions) and control-based WLI applications that run on both Administration and Managed Servers.
- Components such as custom security providers and node managers. A WLI application uses default security providers provided by WebLogic Server. However, your application may use a custom security provider that you have plugged into the WebLogic Server security API. The Node Manager provides high availability to the Managed Servers in your cluster. Your application may or may not use a custom Node Manager.
- External resources such as databases, firewalls, load balancers, and LDAP servers. You may need to upgrade these resources in order to remain complaint with BEA supported platforms.

What is the WLI Upgrade Process

Before you actually begin the upgrade it is useful to identify all the elements of the WLI environment that require updates. You also need to identify the tools, scripts, templates, and source code that is required to automate the tasks used to create the application environment.

The steps in the upgrade process are as follows:

1. Upgrade Custom Security Providers in all the machines in the domain.
2. Upgrade Node Managers in all the machines in the domain.
3. Upgrade External resources such as Firewalls, Load Balancers, Databases, and LDAP servers. For example, Apache 1.3 should be upgraded to 2.0 and Oracle 8.1.7 should be upgraded to Oracle 9i to function with WebLogic Integration 10.2.
4. Check and compare the supported configurations for WebLogic Integration 10.x, 9.x, and 8.x. Ensure that the configurations are upgraded to match version WLI 10.2 specifications.
5. Upgrade the domain on the machine that hosts the Administration Server. WLI provides tools to upgrade the WLI environment with minimal manual effort. The Domain Upgrade Wizard helps you upgrade the domain to 10.2.
6. Upgrade Managed Servers on all the machines in the domain.

7. Upgrade the Application Project and Source. WebLogic Integration allows you to upgrade using any one of the following methods:
   - Use the Application Upgrade Wizard available in Workspace Studio to import the 8.1.x or 8.5 x applications into the Eclipse workspace and then begin the upgrade process to WLI 10.2.
   - You can alternatively upgrade 8.1.x and 8.5 x applications from the command line using an Ant task. In this method you use an 8.1.x and 8.5 x work file as the source parameter and the Eclipse workspace as the destination parameter.

   You have to use Workspace Studio to upgrade 9.x applications to WLI 10.2 as there is no command line utility or Ant task available to update 9.x applications to WLI 10.2.

   During the upgrade the following components are also updated:

   a. WebLogic Integration 8.1.x and 8.5 x artifacts such as JPD, DTF/XQuery, JCX Controls and JCS files to WebLogic Integration 10.2 standards. All file extensions such as `.jpd`, `.jpf`, `.app`, `.jcs`, `.jcx`, and `.jws` are changed to `.java`. All the JPD, DTF, JCX, and JCS Annotations are also updated to the JSR 175 based Annotation model.

   b. (Optional) Upgrade of XQuery 2002 files to XQuery 2004. You may require to update these files manually.

8. Publish and deploy the upgraded application using the upgraded domain. You can publish and deploy using Workspace Studio or from the command line. For more information on publishing and deploying from the command line, see Building Applications with Ant Build Files.

9. View the Upgrade Process log in Workspace Studio to ensure that the upgrade has been successful according to your specifications.

10. Run the upgraded application to test the success of the upgrade process using the Test Form and the Process Graph tabs.

11. You need to recompile and redeploy the applications once the upgrade is complete.

    Note: Ensure that WebLogic Integration 8.1.x and 8.5 x application process instances are run to completion in the appropriate environment before they are used in WebLogic Integration 10.2 environment.

What Impacts the Upgrade Process

In WLI 9.2, there were several architectural level changes, which were carried forward to WLI 10.2. These changes impact the upgrade process. Table 1-1 lists these changes. For a comprehensive list of new features in this release, see *WebLogic Integration 10.2 Release Notes*.

Note: Table 1-1 does not provide a complete list of new features. As a result of these enhancements, WebLogic Platform 8.1 applications will not be binary-compatible and will require automated or manual changes during an upgrade to WebLogic Integration 10.2.

Table 1-1 What Impacts the Upgrade Process

Enhancement	Description	Version Introduced
Eclipse-based IDE	The Workspace Studio IDE is now based on Eclipse, delivering a software development platform that blends open source and commercial software, and is standards-based. The IDE provides access to core Eclipse features, such as source editing, jUnit test integration, and refactoring. It also includes a robust tool set available from the Eclipse Web Tools Platform (WTP) 1.0 project, including server plug-ins for multiple runtimes. For more information about Eclipse 3.2.2 and Eclipse WTP 1.5.4, see http://www.eclipse.org. In WebLogic Integration 9.2 and higher, the IDE delivers design views for developing JPDs. Additional design views to support web service and Java control development will be provided in the future. **Note:** In February 2005, BEA joined the Eclipse Foundation as a Strategic Developer and Board Member to further its commitment to open source and standards organizations.	WLI 9.2
Apache Beehive 2.0	Workspace Studio 1.1 provides tools to make building applications with Apache Beehive 2.0 easier, including support for: • Java controls—based on Plain Old Java Objects (POJO) architecture. • NetUI—based on Struts, and including Page Flows and JSP tags. Apache Beehive is an open-source programming model designed to simplify J2EE programming tasks and is built on J2EE and Struts. BEA developed Apache Beehive, which evolved from its BEA Workspace Studio product, to provide a simplified development model for all WebLogic applications. For more information about Apache Beehive, see http://beehive.apache.org.	WLI 9.2

Table 1-1 What Impacts the Upgrade Process (Continued)

Enhancement	Description	Version Introduced
Metadata Annotations	The programming model for Web Services, EJBs, Java controls, and Java Page Flows uses the new J2SE 5.0 metadata annotation language (specified by JSR-175). In this programming model, you create a Java file that uses annotations to specify the shape and characteristics of the component. From these annotations, the compiler takes care of generating the required supporting artifacts, including Java source code, deployment descriptors, and so on. The annotations that you can specify include: • Web Service annotations defined by *Web Services Metadata for the Java Platform specification* (JSR-181). For more information, see http://www.jcp.org/en/jsr/detail?id=181. • EJB annotations as defined in the EJBGen Reference in *Programming WebLogic Enterprise JavaBeans*. • Java control and NetUI (Page Flow) annotations as defined by Apache Beehive 2.0. For more information, see http://beehive.apache.org. • WebLogic-specific annotations to support security policy configuration, asynchronous failure and response, conversational Web Service support, and more. For more information, see Programming the JWS File in *WebLogic Web Services: Getting Started*.	WLI 9.2
Web Service Policy Framework	Security and authentication configuration has been enhanced to use the standards-based Web Services Policy Framework (WS-Policy), as described in Configuring Message-Level Security in *WebLogic Web Services: Security*.	WLI 9.2

Table 1-1 What Impacts the Upgrade Process (Continued)

Enhancement	Description	Version Introduced

ALSB Control	WLI supports ALSB control that can invoke ALSB proxies and is deployed as a library on WLS using the `config.xml` file.
	In order to use the ALSB control you need to include a library reference in the `weblogic-application.xml` file as follows:
	`<wls:library-ref>`
	`<wls:library-name>sb-transport-control-10.0</wls:library-name>`
	`<wls:specification-version>10.0</wls:specification-version>`
	`<wls:implementation-version>10.0</wls:implementation-version>`
	`</wls:library-ref>`
	The Application Upgrade supports the addition of this entry to the `weblogic-application.xml` file during upgrade from WLI 8.1.x and 8.5 x or 9.x to WLI 10.2.
		WLI 10.2

XMLBean and XQuery API Standards	WLI supports new standards for XMLBeans and XQuery APIs, as described in XMLBeans and XQuery Implementations.	WLI 9.2

Table 1-1 What Impacts the Upgrade Process (Continued)

Enhancement	Description	Version Introduced
Changes in Directory Structure	WebLogic Server offers the following enhancements to the structure of the WebLogic domain directory: • To improve configuration management and promote XML file validation, WebLogic Server supports the specification of domain configuration data in multiple files, including `config.xml` in the new `domain_name/config` directory. (Here, `domain_name` specifies the domain directory.) In previous releases, the `config.xml` file was the repository for all configuration information. Now, new subdirectories of the config directory maintain configuration modules for diagnostic, JDBC, JMS, Node Manager, and security subsystems. Each configuration file adheres to an XML Schema definition. • Startup and shutdown scripts are maintained in the `domain_name/bin` directory. In previous releases, they were stored in the root directory of the domain. In addition to the structural enhancements to the domain directory, WebLogic Server supports new utilities for managing changes to server configuration. These new tools enable you to implement a secure, predictable means for distributing configuration changes in a domain. For more information, see *Understanding Domain Configuration*.	WLI 9.2

Terminology Used in This Document

We recommend that, before proceeding, you familiarize yourself with the following terminology:

- Compatibility—The capability of an application built using one release or service pack to run in another release or service pack, with or without rebuilding the application.
- DTF—Data Transformation File. DTF files have an extension of `.dtf` and contain definitions of a data transformation that can be invoked from a JPD. From WLI 9.2, all `.dtf` files have a `.java` extension. For more information, see, Building Your First Data Transformation.
- IDE—Integrated Development Environment. This refers to the Workspace Studio development environment based on Eclipse, which is a development platform that blends open source and commercial software, and is standards-based.

- Interoperability—(1) The capability of an application deployed in one release or service pack to communicate with another application that is deployed in a different release or service pack. (2) The capability of BEA WebLogic Platform™ components to communicate with third-party software via standard protocols.
- JCS—Java Control Source file. JCS files have an extension of `.jcs`. For more information, see, http://edocs.bea.com/workshop/docs81/doc/en/workshop/guide/controls/conGettingStartedWithJavaControls.html.

  From WLI 9.2, all `.jcs` files have a `.java` extension.
- JCX—Java Control Extension file. JCX files have an extension of `.jcx`. For more information, see http://edocs.bea.com/workshop/docs81/doc/en/workshop/guide/devenv/conJwiFiles.html.

  From WLI 9.2, all `.jcx` files have a `.java` extension.
- JPD—A Java Process defined in a Process Definition for a Java file. From WLI 9.2, all `.jpd` files have a `.java` extension.
- JSR—A Java Specification Request. For more information, see http://jcp.org/en/jsr/overview. From WLI 9.2, all `.jsr` files have a `.java` extension.
- Migrate—To move an application or domain configuration from a third-party product to a BEA product.
- Upgrade—To upgrade your JPD 8.1 and 9.2 source and related files to JPD 10.2 artifacts.
- XQ—A short form for XQuery in some cases. XQuery files on Weblogic Platform have an extension of `.xq`. They contain only the XQuery. So, the term XQ could refer to the XQ file or the XQuery itself.

CHAPTER 2

The Upgrade Process

This document provides information on upgrading from WebLogic Integration™ 8.1.x or 8.5.x, and 9.x to WebLogic Integration 10.2. Topics discussed include:

- Prerequisites
- Upgrading Your WebLogic Integration Domain to 10.2
- Upgrading Applications to WebLogic Integration 10.2

Prerequisites

Before beginning the upgrade process, go through the *Upgrading WebLogic Application Environments* Guide. This guide describes the procedures to upgrade your application environment to WebLogic 10.2. An application environment includes applications, the WebLogic domains in which they are deployed, any application data associated with the domain, and may include external resources, such as database servers, firewalls, load balancers, and LDAP servers.

Upgrading Your WebLogic Integration Domain to 10.2

The WebLogic 10.2 Upgrade Wizard allows you to upgrade domains created only in WebLogic Integration 8.1.x and higher. The WLI domain upgrade plug-in supports cluster enabled domains.

Domain Upgrade from 8.1.x to 10.2

At a high-level, the steps performed by WebLogic Integration during a domain upgrade are as follows:

- Adds resources to support advanced web services including the `file store`, `WseeFileStore`, and the JMS server, WseeJmsServer, and its associated JMS module.
- Adds Beehive shared library modules to support Workspace Studio product applications.
- Adds shared library modules to support Personalization (`P13n`) applications.
- Adds shared library modules to support WebLogic Platform applications.
- Updates and adds JMS and JDBC resources to support WebLogic Platform applications.
- Removes user-defined applications that have been deployed in the domain for applications updated from WLI 8.1.x or 8.5.x to 10.2. You will have to upgrade the source files and compile and redeploy the applications.
- Removes deprecated applications that have been deployed in the domain.
- Removes the `JWSQueueTransport` EJB, if it is present in the domain.
- Adds WebLogic Personalization (P13n) JDBC data sources and connection pools.
- Adds external Event Generators.
- Updates JMS destinations.
- Does not support binary compatibility for WLI 8.1.x or 8.5.x.
- Adds the `SQLAuthenticator` security provider to the domain.

  Note: The users `portaladmin` and `weblogic` are added to the SQLAuthenticator security provider. You can remove these users from the DefaultAuthenticator security provider after the domain is upgraded.

- Updates the following, if any data source is configured to use the PointBase database:
  - The database is automatically loaded in embedded mode and upgraded to PointBase v5.1.
  - The `pointbase.ini` file is updated to set `database.home`, `documentation.home` and `pbembedded.lic` for PointBase v5.1.
  - The database files are renamed from workshop to `weblogic_eval` and the associated data source JDBC driver URLs accordingly fixed.

– The PointBase related environment settings are carried over to the upgraded domain scripts, `setDomainEnv.cmd` and `setDomainEnv.sh`.

Domain Upgrade from 9.x to 10.2

At a high-level, the steps performed by WebLogic Integration during a domain upgrade are as follows:

- Upgrades Beehive shared library modules to support Workspace Studio product applications.
- Upgrades shared library modules to support Personalization (`P13n`) applications.
- Upgrades shared library modules to support WebLogic Platform applications.
- Applications are retained as part of the domain and are deployed automatically when you start the WLI 10.2 domain.
- Supports binary compatibility for WLI 9.x. The applications created in WLI 9.x can be deployed and need not be re-built in WLI 10.2. For more information, see Compatibility Statement for WebLogic Server.
- Supports in-flight process upgrade from WLI 9.x to 10.2. After the domain upgrade, you must run all the long running processes started in the 9.x domain, to completion in WLI 10.2 domain.

For more information on the domain upgrade process and things you need to keep in mind during upgrade, see *Upgrading a WebLogic Domain.*

A domain created in production mode, opens in development mode when upgraded from WLI 9.2 to WLI 10.2. The work around to update the development domain to a production domain is as follows:

1. After the domain upgrade, edit the `setDomainEnv.cmd` file and set `PRODUCTION_MODE=true`
2. Before starting the server, set `JAVA_VENDOR=Sun` (or edit `setDomainEnv.cmd` to add this after the `set WL_HOME=....` line).

The logic of selecting jrockit/Sun JDK in production mode is defined in the `BEA_HOME`\wlserver_10.0\common\bin\commEnv.cmd file.

Upgrade Modes

The domain upgrade wizard supports the following upgrade modes:

- **Graphical**—For upgrading a domain interactively, the Domain Upgrade Wizard using a graphical user interface.
- **Silent**—You can upgrade a domain silently by specifying upgrade requirements in a file.

The following sections provide instructions for:

- Upgrading a Domain in Graphical Mode
- Upgrading a Domain in Silent Mode

Upgrading a Domain in Graphical Mode

The following sections describe how to upgrade a WebLogic domain using the WebLogic Upgrade Wizard in graphical mode:

- Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade a Domain
- Procedure for Upgrading a WebLogic Domain

Note: The console from which you are running the Upgrade Wizard in graphical mode must support a Java-based GUI. If you attempt to start the Upgrade Wizard in graphical mode on a system that cannot support a graphical display, the invocation fails and an error message is displayed.

Starting the WebLogic Upgrade Wizard in Graphical Mode to Upgrade a Domain

To start the WebLogic Upgrade Wizard in graphical mode and upgrade a WebLogic domain on a Windows platform, choose the Domain Upgrade Wizard option from the BEA program group in the Windows Start Menu:

Start > Programs > BEA > Tools > Domain Upgrade Wizard

Note: You can only use this option if you **do not** have to customize the environment to specify JDBC driver classes.

To start the WebLogic Upgrade Wizard in graphical mode and upgrade a WebLogic domain from a Windows command prompt or on a UNIX platform:

1. Verify that the WebLogic domain is not running.
2. Reviewed the Important Notes About the Domain Upgrade Process.
3. Backup the JMS Store, if applicable.

4. Open a command prompt window (on Windows) or a command shell (on UNIX) and set up the environment as follows:
   - Add the WebLogic Server classes to the `CLASSPATH` environment variable and *`WL_HOME`*`\server\bin` to the `PATH` environment variable, where *`WL_HOME`* refers to the top-level installation directory for WebLogic Server.
     You can use the *`WL_HOME`*`\server\bin\setWLSenv` script to set both variables.
   - If you use JMS JDBC stores:
     - Make sure the JDBC driver classes are added to the `CLASSPATH` environment variable.
     - Start the corresponding database.
5. Execute the following script to upgrade your domains.
   - On Windows: *`WL_HOME`*`\common\bin\upgrade.cmd`
   - On UNIX: *`WL_HOME`*`/common/bin/upgrade.sh`

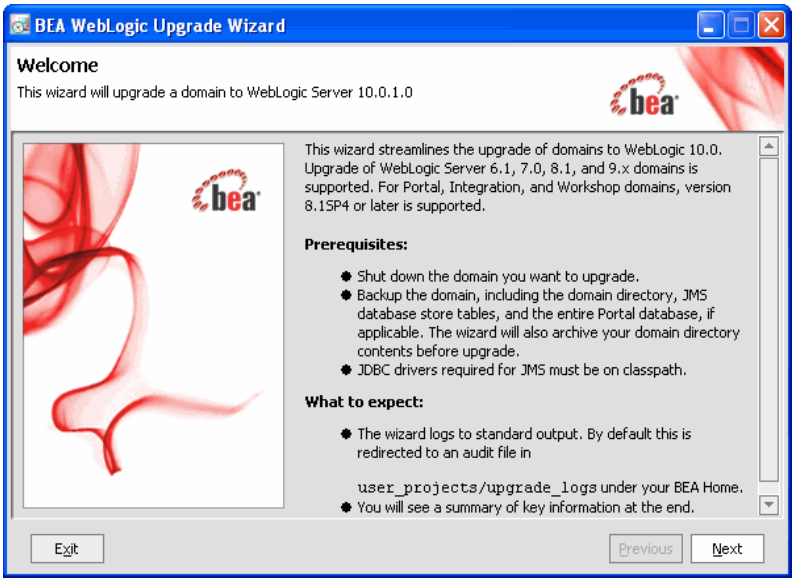
   The log file will be available in the *`BEA_HOME`*`/user_projects/upgrade_logs` directory.

   The following command can also be used to upgrade a domain.

   `java weblogic.Upgrade [-type domain] [-out` *file*`]`

   Two arguments are optional: `-type` and `-out`. Include these arguments if you want to override the default values for the following:
   - The type of upgrade to be performed. If you do not specify a type with the `-type` option, a domain upgrade is performed.
   - The output file in which all standard output (`stdout`) and error messages are written. If you do not specify a file with the `-out` option, such messages are written to the command window, and a summary of messages is displayed at the end of the upgrade process.

   After you run the command, the WebLogic Upgrade Wizard opens, as shown in the following figure.



6. If JMS JDBC stores are used, ensure the corresponding database is running. Note that PointBase databases are automatically started and shut down by the Domain Upgrade Wizard.
7. Click **Next** to proceed to the next window.

Procedure for Upgrading a WebLogic Domain

The following table summarizes the steps in the procedure to upgrade a domain using the WebLogic Upgrade Wizard.

Table 2-1 Procedure for Upgrading a WebLogic Domain

In this step...	You...
Select WebLogic Version	Select the WebLogic version of the domain that you are upgrading. Click **Next** to proceed to the next window.
Select a Domain to Upgrade	Select the directory that contains the WebLogic domain to be upgraded by navigating the local directory hierarchy. Click **Next** to proceed to the next window.

Table 2-1 Procedure for Upgrading a WebLogic Domain (Continued)

In this step...	You...
Inspect Domain	Review progress of the wizard as it inspects the domain. Progress messages are displayed in the window. If you attempt to upgrade a domain in which custom security providers are used, without first upgrading those security providers, an error message is displayed and the wizard exits. If you receive this error message, upgrade the customer security providers, and start the domain upgrade procedure again. Once the inspection is complete (and if no error is encountered), the wizard advances to the next window automatically.
Select Administration Server	Select a server to function as the Administration Server in the new domain. **Note:** If there is only one server defined in the domain, this window is skipped. This window is displayed only if the domain you are upgrading has multiple servers. Click **Next** to proceed to the next window.
Enter Node Manager Credentials	Enter the username and password (and password confirmation) for Node Manager authorization. For WebLogic Server 10.0, Node Manager requires user and password credentials to be specified for each domain. By default, the username and password are set to `weblogic`. If you do not use Node Manager, leave the default values unchanged. Click **Next** to proceed to the next window.

Table 2-1 Procedure for Upgrading a WebLogic Domain (Continued)

In this step...	You...
Select Upgrade Options	• Back up current domain (recommended)—If selected, the wizard backs up the original domain directory and stores it in a zip file. This option is selected by default. **Note:** The wizard backs up the domain directory only and does not preserve file permissions. BEA recommends that you back up the domain and any external application and application database resources in a separate process. • Add log files to backup zip—If selected, log files will be included in the backup zip file. The number and size of log files can be large and you may want to disable this option to exclude them from the backup file. By default, log files are included in the backup file. • Do not set backwards compatibility flags—As of WebLogic Server 9.0, some previously supported behavior has changed to comply with J2EE 1.4. By default, the wizard sets flags to enable the previous behavior in the new domain. If you select this option, these flags are not set for backward compatibility.
Directory Selection Analysis and Optional Tasks	Review progress as the wizard processes the domain information and options provided. Progress messages are displayed in the window. Once processing is complete, the wizard advances automatically to the next window.
Domain Backup	Review progress of the wizard as it prepares to back up the domain. Progress messages are displayed in the window. Once processing is complete, the wizard advances automatically to the next window.
Select Directory for Domain Backup	In this window, set values for the following: • Backup directory — Navigate the local hierarchy and select the directory in which you want to save the backup zip file. By default, the original domain directory is used. • Backup filename—Enter the name of the backup file in the text box. The default filename is `weblogic-domain-backup-`*`domain`*`.zip`, where *`domain`* specifies the name of the domain. Click **Next** to proceed to the next window.

Table 2-1 Procedure for Upgrading a WebLogic Domain (Continued)

In this step...	You...
Backup Domain	Review progress as the wizard backs-up the domain. A progress bar displays the percentage of the backup process that is complete, and progress messages are displayed in the window. **Note:** Backup files created by the wizard need to be protected by the user as they may contain confidential information. Once the backup process is complete, the wizard advances automatically to the next window.
Restructure Domain Directory	Review progress as the wizard restructures the domain directory. Progress messages are displayed in the window. Once the process is complete, the wizard automatically advances to the next window.
Upgrade Configuration Settings	Review progress as the wizard upgrades the configuration settings. Progress messages are displayed in the window. The configuration information is not persisted until a later step. Once the configuration upgrade is complete, the wizard advances automatically to the next window.
Upgrade Persisted Messages and Transaction Log Formats	Review progress as the wizard upgrades the persisted messages (JMS file and JDBC stores) and transaction (JTA) logs that exist in the domain. A progress bar displays the percentage complete and progress messages are displayed in the window. Once the persisted message and transaction log upgrade process is complete, the wizard advances to the next window automatically.
Execute Upgrade of Required WebLogic Personalization Components	Review progress as the Wizard updates Personalization components. Click **Next** to proceed to the next window.
Upgrade RDBMS Authenticator Security Provider	Specify whether or not the deprecated RDBMSAuthenticator should be replaced by the SQLAuthenticator. **Note:** This window appears only when an RDBMSAuthenticator Security Provider exists in the domain you are upgrading. Click **Next** to proceed to the next window.
Prepare WLI Domain Upgrade Plugins	The Wizard will now upgrade WebLogic Integration-specific resources in the domain. Click **Next** to begin the process.

Table 2-1 Procedure for Upgrading a WebLogic Domain (Continued)

In this step...	You...
Execute WLI Domain Upgrade Plugins	Review progress as the Wizard upgrades WebLogic Integration resources in the domain. Click **Next** to proceed.
Finalize Domain Upgrade	Review progress of the wizard as it saves the upgraded configuration and deletes any temporary files that were created during the upgrade process. Progress messages are displayed in the window. **Note:** When upgrading remote Managed Servers, the wizard does not persist the configuration information. Once this process is complete, click **Next** to proceed to the next window.
Database Upgrade Choice	Specify whether or not you would like to upgrade the domain database before proceeding. The Wizard does not back up the domain database. You will need to back up your domain database before beginning a domain upgrade. Select an option and click **Next** to proceed.
Associate DB Categories with Datasources	The table displays the database categories and their associated data sources. The categories are used with their associated data source to initialize the domain database. If a data source appears as undefined, you can update the category with the correct data source. If the data source remains undefined, it will be skipped and not upgraded. In most of the cases, the association is correct and no further changes are required. **Note:** To upgrade the DB category, ensure a data source is associated with it and not left undefined. If the data source remains undefined, it will be skipped and not upgraded. Click **Next** to proceed.
Initialize Category/Datasource Table	**Note:** This window appears only if you have opted to upgrade the domain database. Review progress of the Wizard as it prepares to upgrade the domain database schema objects. The wizard automatically advances to the next window when the process is complete.
Upgrade Database	**Note:** This window appears only if you have opted to upgrade the domain database. Review progress of the Wizard as it upgrades the domain databases. The wizard automatically advances to the next window when the process is complete.

Table 2-1 Procedure for Upgrading a WebLogic Domain (Continued)

In this step...	You...
Upgrade Complete	Review the results of the upgrade, including any important messages that require further consideration. Click **Done** to close the WebLogic Upgrade Wizard.

Upgrading a WLI 10.2 Domain

You can upgrade a WLI domain from WLI 10.2 to WLI 10.2 with WLP 10.2.

WLI 10.2 Domain Without WLP 10.2

1. Install WLI 10.2 without WLP 10.2.
2. Create a WLI domain using the Configuration Wizard. This domain uses WebLogic Server 10.0 MP1 Light Weight Portal Framework libraries.
3. Add the WLP 10.2 installation on top of the WLI 10.2 installation. The WLI 10.2 domain you had created earlier will not work with this new installation as it contains new Light Weight Portal Framework libraries. However, if you create a new domain now, it works with this installation without a problem.
4. Run the domain upgrade on the WLI domain that you created earlier. Confirm that the config.xml file has been updated with 10.2 Light Weight Portal Framework references.

WLI 10.2 Domain With WLP 10.2

If you install both WLI 10.2 and WLP 10.2 and then create a new domain, you do not encounter any problem with the domain.

Upgrading a Domain in Silent Mode

Note: Only WebLogic Server domains can be upgraded in the Silent mode.

In some circumstances, for example, when the domain resides on a remote machine, it is not practical to use the WebLogic Upgrade Wizard in graphical mode. In such situations, you can use the wizard in silent mode to upgrade the WebLogic domain.

To start the WebLogic Upgrade Wizard in silent mode and upgrade a WebLogic domain:

1. Verify that the WebLogic domain is not running.

2. Reviewed the Important Notes About the Domain Upgrade Process.
3. Open an MS-DOS command prompt window (on Windows) or a command shell (on UNIX) and set up the environment as follows:
   - Add the WebLogic Server classes to the `CLASSPATH` environment variable and *`WL_HOME`*`\server\bin` to the `PATH` environment variable, where *`WL_HOME`* refers to the top-level installation directory for WebLogic Server.
     You can use the *`WL_HOME`*`\server\bin\setWLSenv` script to set both variables.
   - If you use JMS JDBC stores:
     - Make sure the JDBC driver classes are added to the `CLASSPATH` environment variable.
     - Start the corresponding database.
4. (Optional) Create an XML script to define the upgrade requirements. For more information see Silent Upgrade XML Script Reference.
5. Navigate to the directory that contains WebLogic domain that you want to upgrade. For example:

   ```
   cd c:\bea\user_projects\domains\domain
   ```

   where *`domain`* specifies the name of the domain.
6. At a command prompt, enter the following command:

   ```
   java weblogic.Upgrade -mode silent -type domain [-responses xmlfile]
   [-out file]
   ```

   The following arguments are optional: `-responses` and `-out`. Include these arguments if you want to override the default values for the following:
   - The location of an XML file that defines the upgrade requirements. If you do not specify a file with the `-responses` option, the wizard uses the default values during the upgrade process. For more information about the format of the XML file and the default values, see Silent Upgrade XML Script Reference.
   - The output file in which all standard output (`stdout`) and error messages are written. If you do not specify a file with the `-out` argument, these messages are written to the command window.

Known Limitations for Domain Upgrade

When you are upgrading stateful JPD applications from WebLogic Integration 9.x you could encounter the following error:

```
java.io.InvalidClassException: javax.xml.namespace.QName; local class
incompatible: stream classdesc serialVersionUID = 4418622981026545151,
local class serialVersionUID = -9120448754896609940
```

This issue is due to a known bug in the JDK.

After upgrading the domain, before you restart the server for the upgraded domain, the suggested solution for systems running on:

- Windows is as follows:

  Add the flag `-Dcom.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0` to the `JAVA_OPTIONS` variable in the `startWeblogic.cmd` file, located under the domain_home\bin directory to ensure a successful running process. Modify `set JAVA_OPTIONS=%SAVE_JAVA_OPTIONS%`

  to

  ```
  set JAVA_OPTIONS=%SAVE_JAVA_OPTIONS%
  "-Dcom.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0"
  ```

- UNIX and Linux is as follows:

  Modify the `SAVE_JAVA_OPTIONS="${JAVA_OPTIONS}"`

  to

  ```
  SAVE_JAVA_OPTIONS="${JAVA_OPTIONS}
  -Dcom.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0"
  ```

Upgrading Applications to WebLogic Integration 10.2

WebLogic Integration 10.2 provides a set of utilities that allow you to upgrade your 8.1.x, 8.5.x, and 9.x applications to 10.2. This section describes how to upgrade applications built using WebLogic Integration.

Note that during upgrade, the logic and intent of the application is not altered. WebLogic Integration simply migrates the code to make it compatible with 10.2. This would involve changes like making the applications compatible with the Eclipse Framework and converting Javadoc annotations to JSR-175 compliant annotations, among others.

Before You Begin

Verify that you have completed the following tasks:

- Migrated all your applications to 8.1 (SP4, SP5, or SP6) or to 8.5 (or higher). For information on upgrading your older applications to these versions, see *WebLogic Integration 8.1 Upgrade Guide*.
- Un-deployed all version 8.1 applications before you upgrade the server.
- Verified that the WebLogic domain is not running.
- Check out your version 8.1.x or 8.5.x, or 9.x application sources that need to be upgraded to 10.2.
- Upgraded the WebLogic Integration domain using the WebLogic Platform Domain Upgrade Wizard. For more information on upgrading your domain, see *Upgrading a WebLogic Domain.*
- Use deployment plans to customize the generation of the EAR files and specify the following tuning parameters:
  - `<transaction-isolation-level>`
  - `<transaction-timeout>`
  - `<message-transaction-timeout>`
  - `<ejb-concurrency-strategy>`
  - `<web-tier-controls>`
  - `<async-dispatch-policy>`

  For more information, see Configuring Applications for Production Deployment in *Deploying Applications to WebLogic Server.*

The Upgrade Process

Application upgrade is a three-step process: going through a list of items that will be upgraded, performing the application upgrade, and fixing errors reported in the log to ensure your applications run in WLI 10.2 without any problems.

You can choose to upgrade your user applications using the Import Wizard or the Command Line utility - both provided by Workspace Studio. Alternatively, you could use an Ant task.The subsequent sections describe these methods.

Notes:

- The upgrader does not support upgrade of user-developed helper source files such as Helper classes and 7.x controls.
- If you have specified any custom classloader hierarchies in addition to the standard classloader inversion hierarchy enforced by the 8.1.x or 8.5.x process application, the application upgrader will not recognize these hierarchies. Instead, it generates a standard classloader inversion hierarchy that a WebLogic Integration 10.2 process application requires. You will then need to re-create your custom class loader hierarchy after the application upgrade is complete and then specify the Classloader hierarchy in the `weblogic-application.xml` file.
- To use ALSB control with WLI 10.2, you must add a library reference for the control in the `weblogic-application.xml` file as follows:

  ```
  <wls:library-ref>
      <wls:library-name>sb-transport-control-10.0</wls:library-name>
      <wls:specification-version>10.0</wls:specification-version>
      <wls:implementation-version>10.0</wls:implementation-version>
  </wls:library-ref>
  ```

  The Application Upgrade allows you to add this entry to the `weblogic-application.xml` file during upgrade from any WLI version to WLI 10.2.
- Handler methods can only throw throwables that are declared by Interface methods. This indicates that if the eventset methods declaration throws a exception then its implementation should also throw a proper exception. On the other hand, if the eventset methods declaration does not throw an exception, then its implementation clause also should not throw an exception. Please note that this was not handled by the WLI 8.x compiler, so if an WLI 8.x application is upgraded with improper throwables, it would lead to a compilation error in WLI 10.x. The workaround for this problem is to add or remove the throwable in accordance with the interface definition.

  In this example, the eventset declaration for Callback method onTaskCompleted and also its implementation in AdvanceTaskCtrlImpl.java throws an exception which is as expected.

```
BasicTaskCtrl.java
-----------------------------------------
@ControlExtension()
@com.bea.control.TaskAnnotations.TaskCreate(taskPlanId =
```

```
@com.bea.control.TaskAnnotations.TaskPlanID(path =
'/Worklist/Compatibility 8.1.x', version = 9.0f,
worklistHostApplicationId = 'worklist-ejbs-81x'),
isControlLevelAnno = true)
public interface BasicTaskCtrl extends TaskControl
{
...................
    @EventSet()
    public interface Callback extends TaskControl.Callback {
/**
* @jc:task-event event-type='complete' response='{response}'
*/
@com.bea.control.TaskAnnotations.TaskEventAnno(eventType =
com.bea.wli.worklist.api.events.data.TaskEvent.Type.COMPLETE,
responseParamName = '{response}')
void onTaskCompleted(WliBaseWFTODocument response)throws Exception;
    }
}
AdvanceTaskCtrlImpl.java
-----------------------------------------
public class AdvanceTaskCtrlImpl implements AdvanceTaskCtrl,
ControlSource
{
   ...................
   /**
     * @common:control
     */
    @org.apache.beehive.controls.api.bean.Control()
    private cn.ccb.clpm.wli.control.BasicTaskCtrl basicTaskCtrl;
    @EventHandler(field = 'basicTaskCtrl',
                  eventSet = BasicTaskCtrl.Callback.class,
                  eventName = 'onTaskCompleted')
```

```
    public void basicTaskCtrl_onTaskCompleted(WliBaseWFTODocument
response)
throws Exception
    {
       ..................
    }
}
```

Upgrading the 8.1.x or 8.5.x Application Using the Import Wizard

You can use the Import Wizard provided by Workspace Studio to upgrade your applications to 10.2. The wizard does not alter the logic and intent of the existing 8.1.x or 8.5.x application, nor extract the application from any source repository. It migrates the 8.1.x or 8.5.x source artifacts into the 10.2 source and project model. However, it retains the 8.1.x or 8.5.x Javadoc annotations as they do not require any special processing in 10.2. These annotations are also retained to facilitate any manual processing that may be required after upgrading the application.

The Import Wizard executes the following tasks:

- Imports upgraded source code to the WebLogic Integration 10.2 workspace that you have defined.
- Upgrades 8.1.x or 8.5.x annotations to WebLogic Integration 10.2.
- Migrates your WebLogic Integration 8.1.x or 8.5.x source artifacts to WebLogic Integration 10.2. This involves the following steps:
  - Converts WebLogic Integration 8.1.x or 8.5.x project types to WebLogic Integration 10.2.
  - Optionally moves libraries from the 8.1.x or 8.5.x application Libraries folder to a new EAR project in the upgraded application.
  - Moves JSP files into a WebContent directory.
  - Upgrades Beehive NetUI JSP tags to WebLogic Integration 10.2.
  - Optionally migrates Beehive NetUI JSP tags to Apache Beehive JSP tags.
  - Moves XSD files that are in a Schema project into a Schemas folder of the Utility project.
  - Moves Java packages and source into a `src` directory.

Note: When you upgrade an 8.1.x or 8.5.x application with an EJB or non-web or non-utility project that uses JPD or Process Proxy to make an RMI call to the business process, do not add a process facet to all the non-web or non-utility projects. Instead, add the Library (Process Libraries) to the project's `java` build path as follows:

- Select **Project > Properties > Java build**.
- Select the Libraries tab, click **Add Library,** and select **Process Libraries**.

Note: When you build the application after upgrade, the Eclipse log contains an InvocationTargetException for an application upgraded from WLI 8.1.x or 8.5.x to 10.2. The steps to re-create this exception are as follows:

- Import a WLI 8.1.x or 8.5.x application.
- After the upgrade is successful, check the eclipse log. In some machines, you will find a InvocationTargetException in the Eclipse log.

The steps to remove this exception are as follows:

- Select a new workspace.
- In Workspace Studio, select **Windows > Preferences > Validation** and clear the XML validator check box.
- Repeat the upgrade process on the same application. The exception does not appear in the Eclipse log.

For a step by step guide to upgrade an 8.1.x or 8.5.x application to 10.2, see Upgrading the 8.1 Application Source.

Upgrading the 9.x Application

During the upgrade from 9.x to 10.2, the IDE updates project metadata, moves your facets to version 10.2, and requires a version 10.0 server. The project files are updated to WLI 10.2 and not the source.

The key changes during the upgrade are as follows:

- The runtime is upgraded to 10.x.
- All the facets versions are upgraded to 10.x.
- Factory paths and classpaths are reconfigured to fit in to 10.x
- Libraries/classpath containers are upgraded to 10.x

For a step by step guide to upgrade an 9.x application to 10.2, see Upgrading the 9.2 Application Source.

Note: Once the upgrade to 10.2 is complete, you must redo any manual changes you made to the 9.x project before the upgrade.

Using the Command Line to Upgrade 8.1.x or 8.5.x Applications

Workspace Studio also provides a command line utility that converts the entire 8.1.x or 8.5.x application to work with WebLogic Integration 10.2.

The utility does not check out or delete files. It also does not check in the newly upgraded files automatically. It just copies the essential files over to the WLI 10.2 workspace for migration.

Note: When you run the command line utility, use a 1.5 implementation of the JRE. Ensure that the classpath includes `<%ECLIPSE_HOME%>/startup.jar`.

The command to upgrade your application is as follows:

```
java –cp %ECLIPSE_HOME%/startup.jar
-Dwlw.application=%WORK_FILE%
-Dweblogic.home=%WL_HOME%
org.eclipse.core.launcher.Main
-application com.bea.workshop.upgrade81.upgradeStarter
-data %WORKSPACE%
-pluginCustomization %PREFS_FILE%
```

where,

`ECLIPSE_HOME`	Refers to the path to the directory containing the `startup.jar`. The default for Workspace Studio is: `BEA_HOME/workshop_10.2/workshop4WP/eclipse`
`-Dweblogic.home=WL_HOME`	Refers to the location of WebLogic Server root folder. By default, this is: `BEA_HOME/wlserver_10.0`
`-Dwlw.application=WORK_FILE`	Refers to the application that requires the upgrade. Replace `WORK_FILE` with the work file name corresponding to the WebLogic Workshop 8.1 that you want to upgrade.
`-application com.bea.workshop.upgrade81.upgradeStarter`	Refers to the Eclipse plug-in extension point used to execute this command.

`-data WORKSPACE`	Refers to the name of the target workspace where you want the upgraded application to go. This can be any directory in which you want the version 10.2 application files generated.
`[-pluginCustomization PREFS_FILE]`	Specifies a properties file used to set options for the upgrade. Replace the `PREFS_FILE` with the name of a properties file containing a number of key-value pairs. The possible properties are: • `application` refers to the plug-in extension point to execute at run time. • `weblogic.home` refers to the location of the WebLogic Server root folder. • `data` refers to the name of the target workspace where the upgraded application resides. The name of the parameter is provided by Eclipse and it cannot be overwritten. • `wlw.application` refers to the name of the application work file. • `pluginCustomization` refers to the name of a properties file containing a number of key-value pairs.
Optional Parameters	
`com.bea.wlw.workshop.upgrade81/upgradeHarnessAbortOnError=true/false`	If you do not specify this attribute, the default is `false`. In this case, the upgrader tries to continue after an error. When it is set to `true`, the upgrade process fails when it encounters any error. These errors are listed in the log file.
`com.bea.wlw.workshop.upgrade81/upgradeHarnessMessageLevel`	This attribute indicates a message level. If you do not specify this attribute, the upgrader logs all messages. You can specify the following values for this attribute: • `INFO`: Displays all messages. • `WARNING`: Displays warning, error, and fatal messages,and suppresses informational messages. • `ERROR`: Displays only error and fatal messages.
`com.bea.wlw.workshop.upgrade81/migrateJSPPreference=true/false`	If you do not specify this attribute, the default is `false`. When it is set to `true`, the upgrade process migrates the JSP files to their new Beehive annotation.
`com.bea.wlw.workshop.upgrade81/useJ2EESharedLibraries=true/false`	When you set this attribute to `false`, the upgrade copies the web application libraries to WEB-INF/lib. The upgrade uses J2EE shared libraries by default.

com.bea.wlw.workshop.upgrade81/upgradeHarnessReportOnly=true/false	When you set this attribute to `true`, the upgrade report is generated. The default setting is `false`, and with this setting both the report and upgrade are performed.
com.bea.wlw.workshop.upgrade81/upgradeHarnessLogFile=<log file location>	Use this attribute to specify the location of the upgrade log file. The default value is <workspace location>/.metadata/upgrade.log
com.bea.wlw.workshop.upgrade81/upgradeProjectImportOverwrite=true/false	Use this attribute to specify whether an existing project is overwritten in the event of a conflict in project name. The default value is `false`.
com.bea.wlw.workshop.upgrade81/upgradeProjectImportPrefix	Use this attribute to specify an optional prefix to append to all imported projects.
com.bea.wlw.workshop.upgrade81/upgrade rPrefMoveResourceBundle = true/false	Use this attribute to specify whether files with the `.properties` extension are copied or moved from the web content folder to the source file folder. The default value is `false`.

Note: To upgrade 9.x applications to WLI 10.2, you have to use Workspace Studio, as there is no command line utility available to upgrade 9.x applications to WLI 10.2. The metadata is upgraded when you open the project in the IDE.

Using an Ant task to Upgrade 8.1.x or 8.5.x Applications

You can use the Ant task to upgrade from WLI 8.1.x or 8.5.x to WLI 10.2.

The command line upgrade contains an Ant task. You can locate the class of the Ant task in the `wlw-upgrade.jar`, deployed in the `%BEA_HOME%/tools/eclipse_pkgs/1.1/pkgs/eclipse/plugins/com.bea.workshop.upgrade81_1.0.20` folder.

Note: When you run the Ant task, ensure that the `<%ECLIPSE_HOME%>/startup.jar` is on the classpath of the task, as specified by the `classpathref` attribute in the following sample Ant task.

A following sample shows the content of the Ant task (upgrade.xml):

```
<!-- Upgrade.xml : Target to upgrade the 8.1.x or 8.5.x app to Darjeeling
-->.
```

```
<target name="workshopUpgrade">
<echo message="Upgrading 8.1.x or 8.5.x located at ${WORK_FILE} to
${WORKSPACE}" />
<path id="eclipse.classpath">
<pathelement path="${env.CLASSPATH}" />
<fileset dir="${PKGS.HOME}/eclipse/plugins/"
includes="com.bea.workshop.**/wlw-upgrade.jar" />
</path>

<taskdef name="upgradeTask"
classname="com.bea.workshop.upgrade81.cmdline.UpgradeTask"
classpathref="eclipse.classpath" />
<upgradeTask data=%WORKSPACE%
eclipseHome=%ECLIPSE_HOME%
weblogicHome=%WL_HOME%
pluginCustomization=%PREFS_FILE%
wlwApplication=%WORK_FILE%/>
</target>
```

The following example shows how you can invoke the Ant task to perform an application upgrade from the command line:

```
ant -f Upgrade.xml workshopUpgrade -DWORK_FILE=C:\xyz\allWorkflows3.work
-DWORKSPACE=C:\tempcmdupgrade
```

where,

`WORKSPACE`	Refers to the Eclipse workspace that the WebLogic Integration 8.1.x or 8.5.x application is imported and upgraded to.
`ECLIPSE_HOME`	Refers to the Eclipse directory containing the `startup.jar`. `<%BEA_HOME%>\tools\eclipse_pkgs\1.1\eclipse_3.2.2`

PKGS.HOME	Refers to the location of the Eclipse packages. <%BEA_HOME%>\tools\eclipse_pkgs\1.1\pkgs
WL_HOME	Refers to the location of the root folder of WebLogic Server. <%BEA_HOME%>\wlserver_10.0
PREFS_FILE	Refers to the location of an optional preference file used during import or upgrade.
WORK_FILE	Refers to the location of the work file for the WebLogic Workshop 8.1.x or 8.5.x application to be imported or upgraded.

Note: There is no Ant task available to upgrade applications from WLI 9.x to WLI 10.2.

Understanding the Upgrade Log

WebLogic Integration 10.2 generates a log of the upgrade changes, errors, and warnings, irrespective of the upgrade process you choose. If you use the wizard, this log is displayed in a dialog that you can review before the process is complete.

The log file is generated after the upgrade is completed and it is saved as:

```
UPGRADE_WORKSPACE_HOME\.metadata\upgrade.log
```

A log message in the file appears as follows:

```
!SUBENTRY 1 com.bea.workshop.upgrade81 severity_level date time
!MESSAGE Upgrade-related message.
```

The severity level contains two numbers with the same meaning. The date and time entries refer to when the upgrade was attempted. The upgrade-related message describes what was done, warned about, or the error that occurred. The following is a snippet containing two log entry examples:

```
!SUBENTRY 1 com.bea.workshop.upgrade81 2 2 2006-02-27 17:17:53.687
!MESSAGE The 9.2 control context only supports a subset of the 8.1 control
context APIs. Please see the Workspace Studio upgrade documentation for more
information.
!SUBENTRY 1 com.bea.workshop.upgrade81 1 1 2006-02-27 17:17:53.687
!MESSAGE The import "com.bea.control.JwsContext" needs to be updated.
```

Outages During or After Deployment

You might encounter certain outages while trying to deploy your upgraded application. For information on outages, see the "Known Limitations" section, in WebLogic Integration Release Notes.

Manual Changes You Need to Do After Upgrade

- After upgrading the 8.1.x or 8.5.x domain, ensure that you have set the security policies on the Compatibility 8.1.x or 8.5.x Task Plan and enabled the 'Anonymous' role in the Create Policy. Use the Worklist Administration Console (the default authorization provider) to set the Create Policy for the Compatibility 8.1.x or 8.5.x task plan. If you are using a third-party authorizer, use the related third-party client tools to set the policy.
- If you are directly using MFL-derived XMLBeans types for internal use or during conversion of data from non-XML to XML as an intermediate form, you need to manually specify namespaces in the element constructors of these XQuery transformations upgraded from 8.1.x or 8.5.x.
- Some worklist 8.x applications are backward compatible. However, a few of the APIs used in the 8.x application may be deprecated. The upgrade log will contain errors and warnings listing the deprecated APIs in the application. You need to redesign the Task Plans in the upgraded worklist application to experience the benefits of the new worklist features.
- You must regenerate task controls if there is any change to the Task Plan.
- Several worklist 8.x control methods have been removed from the worklist 9.2 task controls. Although the upgrade completes successfully, these methods will be listed as deprecated in the process log file. You must manually redesign the deprecated control methods using the new task control methods.

Testing the Upgrade

After the upgrade is complete, you can optionally build and deploy the upgraded application to verify if the upgrade is successful. You can ensure that the required files have been moved or are available in the correct locations as follows:

- JPD Annotation Processor:
  - The project and component beans that JPD requires must be available in the `build/EJB` directory.

– The `wli-process.xml`, `wli-subscriptions.xml`, and `wlw-manifest.xml` should be available in the `build/processoutput/WEB-INF/`directory.

Note: The `wlw-manifest.xml` file provides information about the server resources referenced in an EAR file. Server administrators should examine the `wlw-manifest.xml` file to determine the resources necessary for successful deployment.
In 8.x, the `wlw-manifest.xml` file was generated in `META-INF/wlw-manifest.xml` of the enterprise application. In 10.x, this file is generated in the WEB-INF directory of each web application.

- Channel Builder contains the `wli-channels.xml file` in the `UtilProject\build\classes\channeloutput` directory.

Upgrade Defects Fixed Since WLI 9.2

Table 2-2 lists the upgrade defects that have been fixed since WLI 9.2.

Table 2-2 Defects Fixed Since WLI 9.2

CR Number	Description
WebLogic Server	
CR262360	Support for 8.x callbacks. Service Control callbacks to JWS from JPD are supported with this fix.
CR294193	Internal weblogic.apache.* classes were incorrectly documented as public. The weblogic.apache.xerces.* classes have been deprecated since WebLogic Server release 9.1. Please use org.apache.* classes instead.
CR256082	WLS Dante JWS is enabled to support Workshop 8.1 JWS-style callbacks. This fix enables WLS Dante Web Services to work with WLI Dijon JPDs, in upgrade scenarios.
CR290303	The rmic compiler of Sun™ ignores manifest classpaths.

CR Number (Continued)	Description

CR282924	Ensuring Correct Handling of `xs:anyType` in Messages
	
	If you created a version 8.1 web service by generating it from a WSDL that specified `xs:anyType` instead of `xs:any`, the web service will expect and send incorrect XML payloads after upgrade to version 9.2. You can ensure correct handling of `xs:anyType` by applying the following annotation to the web service at the class level:
	`@WildcardBindings(`
	
	`@WildcardBinding(className="org.apache.xmlbeans.XmlObject",`
	
	`binding=WildcardParticle.ANYTYPE)`
	`)`

WebLogic Workshop	
CR290412	WLI 8.1 or Portal domains cannot be upgraded if the 9.2 Workshop IDE component is not installed.
CR282622	Support for packaging a third-party control and installing it into a working Workshop/Server installation to ensure that the control is available at design time (in the Workshop environment) and server runtime.
CR276528	Cannot cast from Stack to PageFlowStack after upgrade because (PageFlowStack) PageFlowUtils.getPageFlowStack in an 8.1 application is not upgraded to PageFlowStack.get.
CR278246	An WLI 8.1 custom control with an empty callback interface does not compile after upgrade.
CR280733	The JDBC Control does not support "all" as a value for array-max-length annotation.
CR300285	Add support in Service Control to parse a JMS URL query string for cluster address and set it via stub.
CR291139	Source upgrade fails to handle web.xml containing multiple web-resource-collection element within security-constraints.

CR Number (Continued)	Description
CR292362	Upgraded JWS using MessageBuffer annotation has incorrect values for retryCount and retryDelay.
CR292880	IDE incorrectly indicates “build successful” even though the build is incomplete due to a failure in annotation processing.
WebLogic Integration	
CR299154	Upgraded WLI domain is not configured correctly to support upgraded worklist applications.
CR288777	A B2B application with an ebXML control fails to compile after upgrade because of incorrect package name.
WebLogic Portal	
CR192848	Portal Framework: Switch to JDK 1.5 Internal DOM 3 Parser.
CR291868	Portal Rule Control cannot be invoked from a JPD.

CHAPTER 3

Upgrading Business Processes and Control Files for Use with WebLogic Integration 10.2

The following sections describe updates required to Business Processes and Control files before they can be used with WebLogic Integration 10.2.

Upgrading Business Processes

In the WebLogic Integration 10.2 environment, all JPD files are given a `.java` extension rather than their proprietary extension of `.jpd`. All WebLogic Integration JPD 8.1 annotations are upgraded to JSR 175-based annotations. All the JPD 8.1 or 8.5 annotations are categorized into: common, control and JPD annotations.

In WebLogic Integration 8.1.x or 8.5.x, `jpdContext` within a JPD used to be annotated with `@common:context`. However, in WebLogic Integration 9.2 and higher, `jpdContext` is upgraded to `@com.bea.wli.jpd.Context()`.

For example, a WebLogic Integration 8.1.x JPD Business Process Annotation is as follows:

```
/**
* @jpd:process process::
* <process name="EchoAsync">
*   <clientRequest name="Client Request" method="clientRequest"/>
*   <perform name="Perform" method="perform"/>
```

```
*   <controlSend name="start" method="myTimerStart"/>
*   <clientCallback name="Client Response"
method="clientResponseCallbackHandler"/>
*   <transaction name="Commit"/>
* </process>::
*/
```

After the JPD is upgraded to WebLogic Integration 10.2, the annotation is as follows:

```
@com.bea.wli.jpd.Process(
       process="<process name=\"EchoAsync\">" +
       "  <clientRequest name=\"Client Request\"
       method=\"clientRequest\"/>\n" +
       "  <perform name=\"Perform\" method=\"perform\"/>\n" +
       "  <controlSend name=\"start\" method=\"myTimerStart\"/>\n" +
       "  <clientCallback name=\"Client Response\"
       method=\"clientResponseCallbackHandler\"/>\n" +
       "  <transaction name=\"Commit\"/>\n" +
"</process>"
)
```

Note: `WliTimerControl` is the default WebLogic Integration timer control for JPDs.

Note: JMS transport was supported in WebLogic Integration 8.1 using `jws.queue` for use by Workspace Studio based artifacts (such as business processes and JWS). In WebLogic Integration 10.2, JWS uses `weblogic.wsee.DefaultQueue` as the default queue for JMS transport whereas business processes still require `jws.queue`. Further, note that even though WebLogic Integration 10.2 allows you to specify any JMS queue for JMS transport, you must not use `jws.queue` for new JWS applications as that causes conflict in the WebLogic Integration-enabled domain. Do not use `jws.queue` or customize the queue name (using `jws.properties`) with JWS applications in WebLogic Integration 10.2.

Table 3-1 provides WebLogic Integration JPD 8.1.x or 8.5.x to 10.2 JPD annotation upgrade information.

Table 3-1 JPD Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jpd:ebxml		Ebxml		Specifies the ebXML parameters for a process.
	ebxml-action-mode		ebxmlActionMode	
	ebxml-service-name		ebxmlServiceName	
	protocol-name		protocolName	
jpd:ebxml-method		EbXMLMethod		Specifies the ebXML parameters for a method.
	envelope		envelope	
jpd:mb-static-subscription		MessageBroker.StaticSubscription		Specifies the subscription parameters for a business process.
	channel-name		channelName	
	xquery		xquery	
	filter-value-match		filterValueMatch	
	message-metadata		messageMetaData	
	message-body		messageBody	
	suppressible		suppressible	

Table 3-1 JPD Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jpd:process		Process		Specifies settings for a business process.
	binding		binding	
	name		name	
	freezeOnFailure		freezeOnFailure	
	onSyncFailure		onSyncFailure	
	retryCount		retryCount	
	retryDelay		retryDelay	
	stateless		isStateless	
	process		process	
jpd:rosettanet		RosettaNet		Specifies the Rosettanet properties for a process.
	protocol-name		protocolName	
	protocol-version		protocolVersion	
	pip-name		pipName	
	pip-version		pipVersion	
	pip-role		pipRole	
jpd:selector		Selector		Precedes an XQuery definition in a business process file.
	xquery		xquery	

Table 3-1 JPD Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
`jpd:transform`		`Transform`		Annotates a transformation control instance, which is instantiated automatically at run time.
`jpd:unexpected-message`		`UnexpectedMessage`		Specifies settings that allow a business process to ignore a message received before the process flow encounters the node at which the message is expected.
	`action`		`action`	
`jpd:version`		`Version`		Specifies how to invoke subprocesses when different versions of the parent process exist.
	`strategy`		`strategy`	
`jpd:xml-list`		`XmlList`		Annotates business process variable of Untyped XML - `XmlObjectList`.
`jpd:xquery`		`Xquery`		Precedes the global XQuery definitions in a JPD file.
	`version`		`version`	Represents the version of XQuery language specification.
	`prologue`		`prologue`	

Table 3-1 JPD Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
`jpd:input-message`		`InputMessage`		Validates the typed XBean parameter at run time.
	`validate`		`validate`	

Upgrading JCX or WebLogic Integration Control Files

After the upgrade to WebLogic Integration 10.2:

- All the WebLogic Integration 8.1.x or 8.5.x Control files are renamed with a `.java` extension
- All the WebLogic Integration Control files with 8.1.x or 8.5.x annotations are upgraded to JSR 175 based annotations.
- The Control Interfaces are annotated according to the Beehive standard with `@ControlExtension`.
- New attributes required for any control are added during the upgrade.
- New import statements are added to the existing import statements during the upgrade if required.

For example, if a WebLogic Integration 8.1 JCX contains the following annotation:

```
/**
    * @jc:task-create
    *   name="{name}"
*/
```

In WebLogic Integration 10.2 it is upgraded to:

```
@TaskCreate(name = "{name}",
taskTypeId.path = "/Worklist/Compatibility 8.1.x",
taskTypeId.version = 10.0f,
```

```
taskTypeId.worklistHostApplicationId = "worklist-ejbs-81x"
)
```

Other useful references are:

- Upgrading Controls

Upgrading JCS Control Files

After the upgrade to WebLogic Integration 10.2, JCS control files are renamed with a `.java` extension. The JCS control files that contain WebLogic Integration control annotations are upgraded.

Table 3-2 provides information on upgrades to WebLogic Integration 8.1 to 10.2 JSC annotations.

Table 3-2 JCS Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
`common:xmlns`		`XmlNamespaces.Entry`		All annotations will be child nodes of XMLNamespace.
	`prefix`		`prefix`	
	`namespace`		`namespace`	
`common:target-namespace`		`TargetNamespace`		
	`namespace`		`value`	

Table 3-2 JCS Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
common:security		Security		All annotations will be child nodes of schemas.
	roles-allowed		rolesAllowed	
	roles-referenced		rolesReferenced	
	run-as		runAs	
	run-as-principal		runAsPrincipal	
	single-principal		singlePrincipal	
	callback-roles-allowed		callbackRolesAllowed	
jcs:jc-jar		<none>		Primarily used in WebLogic Workshop 8.1. No longer used in Workspace Studio 1.1.
common:schema		Schemas.Entry		All annotations will be child nodes of schemas.
	file		file	
	inline		inline	
common:message-buffer		MessageBuffer		
	enable		enable	
	retry-count		retryCount	
	retry-delay		retryDelay	

Table 3-2 JCS Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
editor-info:code-gen				Primarily used in WebLogic Workshop 8.1. No longer used in Workspace Studio 1.1.
jcs:control-tags		<none>		Primarily used in WebLogic Workshop 8.1. No longer used in Workspace Studio 1.1.
common:control		Control		The standard Beehive annotation.
common:operation		<none>		No longer needed, because Apache Beehive control framework handles it.
jcs:ide		<none>		Not handled as this annotation belongs to WebLogic Workshop 8.1.x
jc:conversation		Conversation		This annotation is identical to jws:conversation annotation
	phase		value	
jcs:suppress-common-tags		<none>		Primarily used in WebLogic Workshop 8.1. No longer used in Workspace Studio 1.1.

CHAPTER 4

Control Annotations

The following sections describe upgrades to WebLogic Integration Control annotations.

Data Transformation Controls

Table 4-1 provides information on upgrades to Data Transformation Control annotations.

Table 4-1 Data Transformation Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
`dtf:xquery`		`XQuery`		Specifies the global XQuery functions and XQuery namespaces that can be used within the scope of the prologue of the DTF file.
	`prologue`		`prolog`	
	`<none>`		`xqueryVersion`	
`dtf:transform`		`XQueryTransform` `XsltTransform`		Specifies the XQuery and XSLT abstract methods in a DTF file.
	`xquery-ref`		`transformType`	XQueryTransform .TransformMethodType. xquery_ref
	`xquery`		`transformType`	XQueryTransform .TransformMethodType. xquery
	`xslt-ref`		`transformType`	XsltTransform. TransformMethodType. xslt_ref
	`xslt`		`transformType`	XsltTransform. TransformMethodType. xslt
	`<none>`		`value`	Values allowed: xquery-ref or xquery or xslt-ref or xslt

Table 4-1 Data Transformation Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	<none>		xqueryVersion	Only when the xquery-ref or xquery attributes are available.
dtf:schema-validate		SchemaValidate		Specifies if the source parameters or the return value, or both, should be schema-validated.
	return-value		returnValue	
	parameters		parameters	
dtf:xquery-function		XQueryFunction		Specifies that a user-defined Java method (non-abstract) in a DTF file can be invoked from queries.
	<none>		xqueryVersion	

Email Controls

Table 4-2 contains information on upgrades to Email Control annotations.

Table 4-2 Email Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:email		Email		Specifies configuration attributes for the Email control.
	from-address		fromAddress	
	from-name		fromName	
	smtp-address		smtpAddress	
	reply-to-address		replyToAddress	
	reply-to-name		replyToName	
	smtp-username		smtpUsername	
	smtp-password		smtpPassword	
	smtp-password-alias		smtpPasswordAlias	
	header-encoding		headerEncoding	

Table 4-2 Email Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
`jc:send-email`		`EmailControl.Send`		Specifies configuration attributes for the Email control.
	`to`		`to`	
	`cc`		`cc`	
	`bcc`		`bcc`	
	`subject`		`subject`	
	`body`		`body`	
	`content-type`		`contentType`	
	`attachments`		`attachments`	

File Controls

Table 4-3 contains information on upgrades to File Control annotations.

Table 4-3 File Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:file		FileControl.FileInfo		Specifies the annotations for the File control.
	directory-name		directoryName	
	file-mask		fileMask	
	suffix-name		suffixName	
	suffix-type		suffixType	
	create-mode		createMode	
	ftp-username-name		ftpUserName	
	ftp-password		ftpPassword	
	ftp-password-alias		ftpPasswordAlias	
	ftp-host-name		hostName	
	ftp-local-directory		ftpLocalDirectory	

Table 4-3 File Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:file-operation		FileControl.Operation	FileControl.IOOperation	Specifies configuration attributes for a File control.
	io-type		ioType	
	file-content		fileContent	
	record-size		recordSize	
	delimiter-string		delimiterString	
	delimiter-checkbox		delimiterCheckbox	
	encoding		encoding	

HTTP Controls

Table 4-4 contains information on upgrades to HTTP Control annotations.

Table 4-4 HTTP Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:httpsend-data		HTTPSendData		Specifies the URL to which an HTTP message is to be sent, and from which a response is to be received.
	url-name		url	

Message Broker Controls

Table 4-5 contains upgrade information for Message Broker Control annotations.

Table 4-5 Message Broker Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:mb-publish-control		PublishControl.ClassPublish	PublishControl	Defines class level attributes for the Publish control.
	channel-name		channelName	
	message-metadata		metadata	
jc:mb-publish-method		PublishControl.MethodPublish		Defines method level attributes for the Publish control.
	message-metadata		metadata	
	message-body		body	
jc:mb-subscription-control		SubscriptionControl.ClassSubscription	SubscriptionControl	Defines class level attributes for the Subscription Control.
	channel-name		channelName	
	xquery		xquery	
	filter-value-match		classFilterValueMatch	
	<none>		xqueryVersion	Indicates the XQuery Version 2002 or 2004. Upgrade sets the version to 2002 by default.

Table 4-5 Message Broker Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:mb-subscription-method		SubscriptionControl.MethodSubscription		Defines method level attributes for the Subscription Control.
	filter-value-match		filterValueMatch	
jc:mb-subscription-callback		SubscriptionControl.SubscriptionCallback		Defines callback attributes for the Subscription Control.
	message-metadata		metadata	
	message-body		body	

MQSeries Controls

Table 4-6 contains information on upgrades to MQSeries Control annotations.

Table 4-6 MQSeries Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:MQConnectionType		MQControl.Connection		Specifies the connection type for an MQ Series control.
	connectionType		type	
jc:MQConnectionPoolProps		MQControl.ConnectionPool		Specifies the MQ Series connection pool properties for the MQ Series control.
	mqPoolSize		poolSize	

Table 4-6 MQSeries Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:ConnectionPoolTimeout		MQControl.ConnectionPool		Specifies the MQ Series connection pool time-out in seconds.
	conTimeout		timeout	
jc:ConnectionRetrySettings		MQControl.ConnectionPool		Specifies the retry settings for the connection to the MQ Series queue manager.
	retryCount		retryCount	
	retryWaitTimeInMilliseconds		retryWaitTimeInMilliseconds	
jc:MQQueueManager		MQControl.Connection		Specifies the name of the queue manager for connection.
	queueManager		QueueManager	
jc:MQAuthorization		MQControl.Connection		Specifies the MQ Series authorization property for the MQ Series control.
	requireAuthorization		authorization	

Table 4-6 MQSeries Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:TCPSettings		MQControl.TCPSettings		Specifies the TCP connection settings for the MQ Series control.
	host		host	
	port		port	
	channel		channel	
	ccsid		ccsid	
	user		user	
	password		password	
	sendExit		sendExit	
	receiveExit		receiveExit	
	securityExit		securityExit	
jc:SSLSettings		MQControl.SSLSettings		Specifies the SSL settings for the MQ Series control.
	sslRequired		sslRequired	
	twoWaySSLRequired		twoWaySSLRequired	
jc:DefaultQueue		MQControl.Connection		Specifies the default queue name to be used for sending and retrieving messages.
	defaultQueueName		defaultQueueName	

Table 4-6 MQSeries Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
`jc:ImplicitTransaction`		`MQControl.Connection`		Specifies the transaction mode of the MQ Series control.
	`implicitTransactionRequired`		`implicitTransaction`	

Process Controls

Table 4-7 contains information on upgrades to Process Control annotations.

Table 4-7 Process Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
`common:message-buffer`		`MessageBuffer`		Specifies that there should be a queue between the component's implementation code and the message transport connection for the specified method or callback.
	`enable`		`enable`	
	`retry-count`		`retryCount`	
	`retry-delay`		`retryDelay`	

Table 4-7 Process Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
`jc:conversation`		`Conversation`		Specifies the role that a control method or callback plays in a conversation. This annotation is identical to the `jws:conversation` annotation.
	`phase`		`value`	
`jc:location`		`Location`		Specifies the URL at which a Web service control accepts requests for each supported protocol. This annotation is identical to the corresponding web service annotation, `@jws:location`.
	`uri`		`uri`	
	`http-url`		`httpUrl`	
	`jms-url`		`jmsUrl`	

Service Broker Controls

Table 4-8 contains information on upgrades to Service Broker Control annotations.

Table 4-8 Service Broker Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
`common:define`		`Defines.Entry`		Defines in-line data with the component class that might otherwise be referenced as an external file.
	`name`		`name`	
	`value`		`value`	
`jc:conversation`		`Conversation`		Specifies the role that a control method or callback plays in a conversation. This annotation is identical to the `jws:conversation` annotation
	`phase`		`phase`	
`jc:location`		`Location`		Specifies the URL at which a web service control accepts requests for each supported protocol. This annotation is identical to the corresponding web service annotation, `@jws:location`.
	`uri`		`uri`	
	`http-url`		`httpUrl`	Converts to type `String[]`

Table 4-8 Service Broker Controls

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	jms-url		<none>	This attribute is ignored.
jc:wsdl		Wsdl		Specifies a WSDL file that is implemented by a web service.
	file		value	

Task Control-level Annotations

Table 4-9 contains information on upgrades to Task Control-level annotations.

Table 4-9 Task Control-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
N/A		TaskControl.TaskPlanID		Should have @TaskType or @TaskCreate. If no jc:task annotation exists, create a @TaskPlanID annotation.
			path	Hard-coded to /Worklist/Compatibility WebLogic Integration 8.1.x
			version	Hard-coded to WebLogic Integration 9.0
			worklistHostApplicationId	Hard-coded to worklist-ejbs-81x

Table 4-9 Task Control-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:task (control)		TaskAnnotations.TaskCreate		
	<NA>		taskPlanId	Hard-coded value
	Name		name	
	description		description	
	comment		comment	
	Priority		priority	
	Owner		owner	

Table 4-9 Task Control-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:advanced (control)		TaskAnnotations.TaskCreate		
	can-be-reassigned		canBeReassigned	Compatible with WebLogic Integration 8.1x only
	can-be-returned		canBeReturned	Compatible with WebLogic Integration 8.1.x only
	can-be-aborted		canBeAborted	Compatible with WebLogic Integration 8.1.x only
	claim-due-business-date		claimDueDate.businessTime.duration	Compatible with WebLogic Integration 8.1.x only
	claim-user-calendar		claimDueDate.businessTime.isUserCalendar = true and claimDueDate.businessTime.calendarName	Compatible with WebLogic Integration 8.1.x only
	Claim-calendar		claimDueDate.businessTime.isUserCalendar = false and claimDueDate.businessTime.calendarName	Compatible with WebLogic Integration 8.1.x only
	completion-due-business-date		completionDueDate.businessTime.duration	

Table 4-9 Task Control-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	completion-user-calendar		completionDueDate.businessTime.isUserCalendar = true and completionDueDate.businessTime.calendarName	
	completion-calendar		completionDueDate.businessTime.isUserCalendar = false and completionDueDate.businessTime.calendarName	
jc:assignee (control)		TaskAnnotations.TaskCreate		
	User		assignmentInstructions81x.users	Comma-separated list converted to String[]
	Group		assignmentInstructions81x.groups	Comma-separated list converted to String[]
	algorithm		assignmentInstructions81x.algorithm	String converted to enum

Task Control Method-level Annotations

Table 4-10 contains information on upgrades to Task Control Method-level annotations.

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:task-create		TaskAnnotations.TaskCreate		
	Name		name	
	description		description	
	comment		comment	
	Priority		priority	
	owner		owner	
	can-be-reassigned		canBeReassigned	Compatible with WebLogic Integration 8.1.x only
	can-be-returned		canBeReturned	Compatible with WebLogic Integration 8.1.x only
	can-be-aborted		canBeAborted	Compatible with WebLogic Integration 8.1.x only
	claim-due-business-date		claimDueDate.businessTime.duration	Compatible with WebLogic Integration 8.1.x only
	claim-user-calendar		claimDueDate.businessTime.isUserCalendar = true and claimDueDate.businessTime.calendarName	Compatible with WebLogic Integration 8.1.x only

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	claim-calendar		claimDueDate.businessTime.isUserCalendar = false and claimDueDate.businessTime.calendarName	Compatible with WebLogic Integration 8.1.x only
	completion-due-business-date		completionDueDate.businessTime.duration	
	completion-user-calendar		completionDueDate.businessTime.isUserCalendar = true and completionDueDate.businessTime.calendarName	
	completion-calendar		completionDueDate.businessTime.isUserCalendar = false and completionDueDate.businessTime.calendarName	
	request	@TaskSetRequestResponse81x		
			value	

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	request-mime-type	@TaskSetRequestResponse81x	mimeType	
jc:task-assign		TaskAnnotations.TaskAssign81x		
	User		instructions81x. users	Comma-separated list converted to String[]
	group		instructions81x. groups	Comma-separated list converted to String[]
	algorithm		instructions81x. algorithm	String converted to enum
jc:task-abort		TaskAnnotations.TaskAbort		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x.
jc:task-resume		TaskAnnotations.TaskResume		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x.
jc:task-suspend		TaskAnnotations.TaskSuspend		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x.

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:task-get-response		TaskAnnotations.TaskGetResponse81x		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x.
	<none>		property=Property.Response	
jc:task-get-request		TaskAnnotations.TaskGetRequest81x		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x.
	<none>		property=Property.Request	
jc:task-get-property		TaskAnnotations.TaskGetProperties		
	name		propertyNames	
jc:task-set-property		TaskAnnotations.TaskSetProperty81x		
	name		name	
	value		value	
jc:task-remove-property		TaskAnnotations.TaskRemoveProperties81x		
	name		propertyNames	

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:task-update		TaskAnnotations.TaskUpdate81x		
	name		name	
	comment		comment	
	priority		priority	
	owner		owner	
	can-be-reassigned		canBeReassigned	Compatible with WebLogic Integration 8.1.x only
	can-be-returned		canBeReturned	Compatible with WebLogic Integration 8.1.x only
	can-be-aborted		canBeAborted	Compatible with WebLogic Integration 8.1.x only
	claim-due-business-date		claimDueDate.businessTime.duration	Compatible with WebLogic Integration 8.1.x only
	claim-user-calendar		claimDueDate.businessTime.isUserCalendar = true and claimDueDate.businessTime.calendarName	Compatible with WebLogic Integration 8.1.x only

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	claim-calendar		claimDueDate.businessTime.isUserCalendar = false and claimDueDate.businessTime.calendarName	Compatible with WebLogic Integration 8.1.x only
	completion-due-business-date		completionDueDate.businessTime.duration	
	completion-user-calendar		completionDueDate.businessTime.isUserCalendar = true and completionDueDate.businessTime.calendarName	
	completion-calendar		completionDueDate.businessTime.isUserCalendar = false and completionDueDate.businessTime.calendarName	
	request	@TaskSetRequestResponse81x (one per method for Request)		

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
			property = Property.Request	
			value = <request>	
	request-mime-type	@TaskSetRequestResponse81x (one per method for Request)		
			property = Property.Request	
			mimeType = <mime type>	
	response	@TaskSetRequestResponse81x (one per method for Response)		
			property = Property.Response	
			value = <response>	
	response-mime-type	@TaskSetRequestResponse81x (one per method for Response)		
			property = Property.Response	

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
			mimeType = <mime type>	

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:task-event		TaskAnnotations.TaskEventAnno		
	name		name	
	comment		comment	
	priority		priority	
	owner		owner	
	can-be-reassigned		canBeReassigned	
	can-be-returned		canBeReturned	
	can-be-aborted		canBeAborted	
	claim-due-business-date		claimDueDate	
	completion-due-business-date		completionDueDate	
	claim-user-calendar		claimDueDate	
	claim-calendar		claimDueDate	
	completion-user-calendar		completionDueDate	
	completion-calendar		completionDueDate	
	request		request	
	request-mime-type		requestMimeType	

Table 4-10 Task Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	response		response	
	response-mime-type		responseMimeType	
	completion-due-date		completionDueDate	
	claim-due-date		claimDueDate	

Task Worker Control-level Annotation

Table 4-11 contains information on upgrades to the Task Worker Control-level annotation.

Table 4-11 Task Worker Control-level Annotation

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:task-worker		<none>		Can be ignored

Task Worker Control Method-level Annotations

Table 4-12 contains information on upgrades to Task Worker Control Method-level annotations.

Table 4-12 Task Worker Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:select		TaskBatchAnnotations.TaskSelect		
	assigned-user		assignedUsers	
	assigned-group		assignedGroups	
	claimant		claimants	
	task-id		taskIds	

Table 4-12 Task Worker Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	task-name		taskName	
	comment		comment	
	owner		owners	
	min-priority		minPriority	
	max-priority		maxPriority	
	states		states	Compatible with WebLogic Integration 8.1.x only
	completion-due-date-before		completionDueDateBefore	
	completion-due-date-after		completionDueDateAfter	
	claim-due-date-before		claimDueDateBefore	
	claim-due-date-after		claimDueDateAfter	
	creation-date-before		creationDateBefore	
	creation-date-after		creationDateAfter	
	property-name		propertyValue.name	
	property-value		propertyValue.value	

Table 4-12 Task Worker Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	selector		selectorParamName	Removes enclosing brackets. For example, "{myParam}" becomes "myParam"
jc:task-create		TaskAnnotations.TaskCreate		
	name		name	
	description		description	
	comment		comment	
	priority		priority	
	owner		owner	
	can-be-reassigned		canBeReassigned	
	can-be-returned		canBeReturned	
	can-be-aborted		canBeAborted	
	claim-due-business-date		claimDueDate	
	completion-due-business-date		completionDueDate	
	request		request	
	request-mime-type		requestMimeType	
	claim-user-calendar		claimDueDate	
	claim-calendar		claimDueDate	

Table 4-12 Task Worker Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	completion-user-calendar		completionDueDate	
	completion-calendar		completionDueDate	
	completion-due-date		completionDueDate	
	claim-due-date		claimDueDate	
jc:task-assign		TaskAnnotations.TaskAssign81x		
	user		user	
	group		group	
	algorithm		algorithm	
jc:task-claim		TaskAnnotations.TaskClaim81x		
	enabled		<none>	This attribute was ignored inWebLogic Integration 8.1.x and 8.5.x
	claimant		claimant	
jc:task-return		TaskAnnotations.TaskReturn81x		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x and 8.5.x

Table 4-12 Task Worker Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:task-start		TaskAnnotations.TaskStart81x		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x and 8.5.x
jc:task-stop		TaskAnnotations.TaskStop81x		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x and 8.5.x
jc:task-complete		TaskAnnotations.TaskComplete		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x and 8.5.x
jc:task-abort		TaskAnnotations.TaskAbort		
jc:task-delete		TaskAnnotations.TaskDelete		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x and 8.5.x

Table 4-12 Task Worker Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:task-resume		TaskAnnotations.TaskResume		
jc:task-suspend		TaskAnnotations.TaskSuspend		
jc:task-get-info		TaskAnnotations.TaskGetInfo		
jc:task-get-response		TaskAnnotations.TaskGetResponse81x		
jc:task-get-request		TaskAnnotations.TaskGetRequest81x		
jc:task-get-property-name		TaskAnnotations.TaskGetPropertyNames81x		
	enabled		<none>	This attribute was ignored in WebLogic Integration 8.1.x and 8.5.x
jc:task-get-property		TaskAnnotations.TaskGetProperties		
jc:task-set-property		TaskAnnotations.TaskSetProperty81x		
	name		name	
	value		value	

Table 4-12 Task Worker Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:task-remove-property		TaskAnnotations.TaskRemoveProperties81x		
	name		propertyNames	
jc:task-update		TaskAnnotations.TaskUpdate81x		
	name		name	
	comment		comment	
	priority		priority	
	owner		owner	
	can-be-reassigned		canBeReassigned	
	can-be-returned		canBeReturned	
	can-be-aborted		canBeAborted	

Table 4-12 Task Worker Control Method-level Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	claim-due-business-date		claimDueDate	
	completion-due-business-date		completionDueDate	
	claim-user-calendar		claimDueDate	
	claim-calendar		claimDueDate	
	completion-user-calendar		completionDueDate	
	completion-calendar		completionDueDate	
	request		request	
	request-mime-type		requestMimeType	
	response		response	
	response-mime-type		responseMimeType	
	completion-due-date		completionDueDate	
	claim-due-date		claimDueDate	

Dynamic Transformation Controls

The following table contains information on upgrades to Dynamic Transformation Control annotations.

Table 4-13 Dynamic Transformation Control Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:ddtf		Ddtf		Specifies the XQuery functions that can be used by the queries and the type of encoding used at design time.
	xquery-prologue		xqueryPrologue	
	control-design-time-encoding		controlDesignTimeEncoding	
jc:xquery		XQuery		Specifies the XQuery files and their attributes for XQuery transformations at run time.
	xquery-arg-names		xqueryArgNames	
	validate-parms		validateParms	
	validate-return		validateReturn	
	design-time-encoding		designTimeEncoding	

Table 4-13 Dynamic Transformation Control Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:xslt		Xslt		Specifies the XSL file to be used for the transformation.
	xslt-arg-names		xsltArgNames	

WebLogic Integration JMS Controls

The WebLogic Integration JMS control is an extension of the base JMS control, and its control annotations also apply to the WebLogic Integration JMS control.

Table 4-14 contains information on upgrades to WebLogic Integration JMS Control annotations.

Table 4-14 WebLogic Integration JMS Control Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:jms		JMSControl.JMS		Sets the JMS properties for the control
	receive-correlation-property		receivecorrelationproperty	
	send-correlation-property		sendcorrelationproperty	
	auto-topic-subscribe		autotopicsubscribe	
	receive-selector		receiveselector	
	topic-table-datasource		topictabledatasource	

Table 4-14 WebLogic Integration JMS Control Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
	send-jndi-name		sendjndiname	
	receive-jndi-name		receivejndiname	
	connection-factory-jndi-name		connectionfactoryjndiname	
	receive-type		receivetype	
	send-type		sendtype	
jc:jms-headers		JMSHeader		Set and retrieves values for the JMS message headers.
	JMSCorrelationID		JMSCorrelationID	
	JMSDeliveryMode		JMSDeliveryMode	
	JMSExpiration		JMSExpiration	
	JMSMessageID		JMSMessageID	
	JMSPriority		JMSPriority	
	JMSRedelivered		JMSRedelivered	
	JMSTimestamp		JMSTimestamp	
	JMSType		JMSType	
jc:jms-property		JMSControl.PropertyValue		Sets and retrieves properties of the message.
	key		name	
	value		value	

TIBCO RV Controls

Table 4-15 contains information on upgrades to TIBCO RV Control annotations.

Table 4-15 TIBCO RV Control Annotations

8.1 Annotation	Attribute	10.2 Annotation	Attribute	Comments
jc:Transport		TibcoRV.Transport		
	service		service	
	network		network	
	deamon		deamon	
jc:UseCM		TibcoRV.UseCM		
	usecm		usecm	
jc:CMTransport		TibcoRV.CMTransport		
	cmname		cmname	
	ledgername		ledgername	
	requestold		requestold	
	syncledger		syncledger	

CHAPTER 5

Other Component Changes

This section provides WebLogic Integration 8.1 to 10.2 upgrade information for the following components:

- Control Factories
- XQuery Files
- JPD and Control Callbacks
- JPD Process Language
- DTF Transformation
- Channel Files

Control Factories

WebLogic Integration upgrades only WebLogic Integration controls used as a control factory from a JPD. WLI updates the source as follows:

1. Adds the @com.bea.wli.common.ControlFactory annotation to the control field declaration in the JPD. For example,

```
@com.bea.wli.common.ControlFactory
@ org.apache.beehive.controls.api.bean.Control
private SampleControlExtension sampleControlExtCF;
```

2. Adds a method with the following signature to the upgraded control extension interface.

   ```
   public <Control Extension type> create();
   ```

   For example,

   ```
   public SampleControlExtension create();
   ```

3. If required, adds the `@com.bea.wli.common.ControlFactoryEventHandler` annotation to the event handler method in the JPD. For example,

   ```
   @ com.bea.wli.common.ControlFactoryEventHandler(field =
   "sampleControlExtCF", eventSet = SampleControlExtension.Callback.class,
   eventName = "response")
   public void receive(SampleControlExtension bean, String data)
   ```

Timer control does not have a control extension. In case of timer control used from a control factory, the upgrader creates a TimerControlFactory control extension class in the same package as the JPD.

For non-Weblogic Integration controls used as a control factory from JPD user must take the following steps after upgrade to be able to use the control from the control factory:

1. Add `@com.bea.wli.common.ControlFactory` annotation to the control field declaration in the JPD

2. Add a method with the following signature to the upgraded control extension interface.

   ```
   public <Control Extension type> create();
   ```

   For example,

   ```
   public SampleControlExtension create();
   ```

3. If required, add the `@com.bea.wli.common.ControlFactoryEventHandler` annotation to the event handler method in the JPD. For example,

   ```
   @ com.bea.wli.common.ControlFactoryEventHandler(field =
   "sampleControlExtCF", eventSet = SampleControlExtension.Callback.class,
   eventName = "response")
   public void receive(SampleControlExtension bean, String data)
   ```

XQuery Files

WebLogic Integration upgrades XQuery files through the upgrade of DTF files. The DTF file contains references to XQuery files that are upgraded along with the DTF file. When the XQuery

file is upgraded, WebLogic Integration includes a comment, at the beginning of the file, that indicates that the file belongs to version 2002.

For example, an XQuery file before the upgrade contains the following:

```
{-- Project3/SwitchAssignTransformation.dtf#forAssign2Copy01 --}

xs:boolean( 'false' )
```

The XQuery file after the upgrade contains the following:

```
{-- Project3/SwitchAssignTransformation.dtf#forAssign2Copy01 --}

{-- version=2002 --}

xs:boolean( 'false' )
```

Note: WebLogic Integration displays a warning message in case you select an upgrade action on an XQuery file stating that the file cannot be upgraded.

Caution: The Xquery within the XQuery files are not upgraded to version 2004: they remain in the version of the original file before the upgrade.

JPD and Control Callbacks

WebLogic Integration upgrades control declarations using `@Control` according to the Apache Beehive standard. The JPD callback field is annotated with `@Callback`. The callback interface is annotated with `@CallbackInterface`. The Callback interface declaration remains a part of the JPD definition and extends the `ServiceBrokerControl`.

According to the Apache Beehive standards, WebLogic Integration also annotates control callback handler methods using `@EventHandler()`.

All the methods in the process definition that are referenced from the `<controlReceive\>` XML snippet are annotated during the upgrade with the `@EventHandler` annotation.

Note: Control callbacks can be sent to a JPD only by using `ControlHandle.sendEvent`.

For example, add the following code to the `MyCustomControlImpl.java` file after upgrade:

```
System.out.println("Before sending event to jpd in
MyCustomControlImpl event

handler");
```

```
    ControlHandle controlHandle = context.getControlHandle();
        try {
        Method m =
MyCustomControl.Callback.class.getMethod("response",
XmlObject.class);
        EventRef event = new EventRef(m);
        controlHandle.sendEvent( event, new Object[]{payload});
        }
        catch(Exception e) {
        e.printStackTrace();
        }
```

JPD Process Language

In WebLogic Integration 8.1.x and 8.5.x applications, the entire process language was specified using `@jpd:process`. However, for WebLogic Integration 10.2 the process language is upgraded to `@com.bea.wli.jpd`. The Process annotation has a `process` attribute that contains the entire process language string.

DTF Transformation

When the DTF files are upgraded, they are renamed with a `.java` extension. All the DTF files in WebLogic Integration 8.1 annotations are upgraded to JSR-175-based annotations. All the controls are converted to Apache Beehive controls.

The DTF files in WebLogic Integration 8.1 have similar functions as other WebLogic Integration controls, but they are abstract classes unlike other controls, which are interfaces. The DTF class contain metadata-specified methods, and well coded methods that are specified by actual Java method bodies that are called by the XQuery engine.

DTF annotations that contained `xquery` and `xquery-ref` attributes indicating XQuery version 2002 have a new `xqueryVersion` attribute in WebLogic Integration 10.2.

WebLogic Integration 10.2 upgrades all import statements and adds new import statements where required. For example, a WebLogic Integration 8.1 DTF file that contains an annotation is as follows:

```
/**
```

```
* @dtf:transform xquery-ref="switchXqAssign2defaultAssign_1Copy01.xq"
*/
```

When this DTF file is upgraded to WebLogic Integration 10.2, it is as follows:

```
@XQueryTransform(value = "switchXqAssign2defaultAssign_1Copy01.xq",
transformType = XQueryTransform.TransformMethodType.xquery_ref,
@com.bea.wli.common.XQuery(version=
com.bea.wli.common.XQuery.Version.v2002)
```

Channel Files

There are no changes to the definition of Channel files. Channel files do not get upgraded during the upgrade process. However, they are moved into the Utility project in the WebLogic Integration application.