BEA WebLogic Portal®

Upgrade Guide

Version 9.2 Maintenance Pack 2
Revised: June 2007

Contents

A. Functional Changes Affecting Your WebLogic Portal Environment

B. Performing Database Upgrade Tasks Manually

CHAPTER 1

Overview of the Upgrade Process from WebLogic Portal 8.1 to 9.2

This section provides an overview of the strategies and procedures for upgrading BEA WebLogic Portal® 8.1 SP4, SP5, and SP6 applications to WebLogic Portal 9.2. The following topics are discussed:

- Definitions
- Portal Version 8.1 SP4, SP5, and SP6 to 9.2 Upgrade Process Overview
- Comparing Supported Features

Much of the WebLogic Portal upgrade is performed by running the WebLogic Upgrade Wizard. The WebLogic Upgrade Wizard described in *Upgrading WebLogic Application Environments*.

Definitions

To clarify the different activities described by this document, a brief list of terms is included:

Migration

Moving an application/domain from a third-party technology to a BEA product. (For example, migrating a customer from IBM, webMethods or "home grown" to BEA.)

Upgrade

Updating BEA platform (and components) from older release/Service Pack to newer release/Service Pack. This includes updating existing application/domain to run in a newer version, for example, 8.1 SP4, SP5, or SP6 to 9.2.

The process required to upgrade an application environment depends on the scope of the application. An application environment includes a WebLogic domain and any applications and application resources associated with the domain. It may also include external resources, such as firewalls, load balancers, databases, and LDAP servers.

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 WebLogic Platform components to communicate with third-party software using standard protocols.

Compatibility

Application built using one release/Service Pack running in another release/Service Pack. This may or may not involve rebuilding the application.

Portal Version 8.1 SP4, SP5, and SP6 to 9.2 Upgrade Process Overview

WebLogic Portal enables you to easily upgrade your 8.1 SP4, SP5, and SP6 applications to 9.2. With the exceptions noted in this guide, WebLogic Portal APIs have been maintained in WebLogic Portal 9.2, and most core formats for the database and file based assets have not changed. Where changes have been made, tools are provide to upgrade you to the new format, or make you aware of manual changes where those are needed.

The upgrade process involves upgrading WebLogic Portal 8.1 SP4, SP5, and SP6 portal applications and resources to WebLogic Portal 9.2. The high level steps in the upgrade process are:

1. Upgrade your domain using the WebLogic Upgrade Wizard. For more information, see "Upgrade Steps" on page 2-2.
2. Upgrade existing WebLogic Portal Version 8.1 SP4, SP5, and SP6 applications to run in WebLogic Portal Version 9.2. You can do this automatically using the Import utility that is provided in WebLogic Workshop. For instructions on using this utility, refer to the Upgrading WebLogic Portal Projects from Version 8.1 to Version 9.2 chapter of the Portal Development Guide.

Comparing Supported Features

This section outlines significant feature changes between the WebLogic Portal 8.1 and the WebLogic 9.2 release.

Webflows and Pipelines Not Supported in Version 9.2

Webflows and pipelines were deprecated in Version 8.1 and are no longer supported; you can use page flows in place of these deprecated features.

Security

Previous releases of WebLogic Portal included a WebLogic Portal-specific RDBMSAuthenticator. This has been deprecated. WebLogic Server 9.2 contains a new default SQLAuthenticator authentication provider, which contains an RDBMS user store for users and groups. BEA recommends upgrading to the new WebLogic Server SQLAuthenticator.

When you run the WebLogic Upgrade Wizard to upgrade your domain, it determines whether or not you are using the RDBMSAuthenticator in your 8.1 installation. If the RDBMSAuthenticator is detected, the WebLogic Upgrade Wizard prompts you to choose whether to upgrade to the WebLogic SQLAuthenticator as your default authentication provider or continue to use your existing RDBMS user store:

- Upgrade users and groups – Choose to automatically upgrade your users and groups from WebLogic Portal 8.1 to the new RDBMS user store, which is part of the new default WebLogic SQLAuthenticator authentication provider. When you run the WebLogic Upgrade Wizard and it detects the Portal 8.1 RDBMSAuthenticator, you can select the Upgrade RDBMSAuthenticator option. Selecting this option replaces the existing authentication provider with the new SQLAuthenticator authentication provider and upgrades your user store, including users and groups. Your `config.xml` file is also updated.
- Do not upgrade users and groups – Choose to continue to use the RDBMS user store from the default RDBMSAuthenticator in your WebLogic Portal 8.1 installation. When you run the WebLogic Upgrade Wizard and it detects your Portal 8.1 RDBMSAuthenticator, you can select the Do not upgrade RDBMSAuthenticator option. You can choose to manually upgrade your users and groups to the Portal 9.2 RDBMS user store later.

If you do not upgrade your user store during the domain upgrade process, you can perform a manual upgrade later. The script to upgrade from the WebLogic Portal-specific RDBMS Authenticator to the WebLogic SQL Authenticator is
WEBLOGIC_HOME`/common/p13n/db/`*dbms*`/upgrade_fromdbmsauth_towlssqlauth.sql`

For additional information, see “Upgrading PointBase Databases” on page B-3.

Note: If you upgrade a WebLogic Portal 8.1 application to 9.2 and you use the `UserProviderControl.createUser()` class in the upgraded domain, you might see a `javax.security.auth.login.LoginException` error when a new user attempts to

log into WebLogic Portal. This occurs because by default new users in a WebLogic Portal 9.2 domain are created in the SQLAuthenticator and not in a migrated authentication provider (which normally is configured with a JAAS flag set to `REQUIRED`). Since the WebLogic Portal domain upgrade wizard does not adjust your JAAS settings or remove your existing authentication provider, you must change the JAAS setting or delete the authentication provider (as appropriate) to avoid this exception.

Interaction Management (Personalization)

Interaction Management enables you to develop, manage, and measure personalized portal applications. Personalization and Campaign management combine to form the foundation of Interaction Management. These functions help you target content to a desired audience.

When you run the BEA WebLogic Upgrade Wizard, the wizard upgrades your WebLogic Portal 8.1 interaction features, including Content Selectors, Placeholders, Segments, and Campaigns.

When you run the BEA WebLogic Upgrade Wizard and it detects your Portal 8.1 installation, you can select the Upgrade RDBMSAuthenticator option, as described in "Security" on page 1-3. Selecting this option replaces the existing authentication provider with the new WebLogic Server SQLAuthenticator and upgrades all content, including personalization features. You can also choose to manually upgrade your personalization features to the Portal 9.2 RDBMS user store later.

Note: Events will fire for a content repository that was upgraded to 9.2 (unless you turned event tracking off at the repository level). Events can include repository configuration changes, as well as content additions, updates, and deletions to the repository. Events will not fire for content in an 8.1 repository that was not upgraded. Events will be fired for content that is added, updated, or removed from that repository.

If you created a separate behavior tracking database in version 8.1, upgrade it as described in "Upgrading Separate Behavior Tracking Databases" on page B-4.

Content Management API Changes

The WebLogic Portal 8.1 content management API has been deprecated and a new content API is introduced for WebLogic Portal 9.2. Although the 8.1 API still functions within 9.2, all new portal development should use the 9.2 API. For more information about the new content API, see the WebLogic Portal JavaDoc.

The 7.0 ContentManager and DocumentProvider APIs were deprecated in WebLogic Portal 8.1. For WebLogic Portal 9.2, they have been completely removed and are no longer supported. It is recommended that you use the 9.2 API.

In addition, the Service Provider Interface (SPI) that is used to connect third-party repositories to the Virtual Content Repository has been extended to allow SPI users to take advantage of new features and the new content API. For more information about the new SPI, see WebLogic Portal JavaDoc.

For a complete list of all the deprecated classes in the API, see WebLogic Portal JavaDoc.

Autonomy

A new Autonomy engine is installed with WebLogic Portal 9.2. It is disabled by default after upgrade so it can be properly configured before launching. Autonomy services that have been enabled in WebLogic Portal 8.1 are also disabled after upgrade. For information about installing and configuring search services, see the WebLogic Portal Search Guide.

In addition, the Autonomy APIs and the all content management APIs, including search, have been deprecated in WebLogic Portal 9.2. If you want to continue to use the deprecated APIs, you need to manually add the older Autonomy APIs to your application, as described in "Using 8.1 Search within a 9.2 Portal Application" on page A-9.

For information about upgrading to the new version of Autonomy that is included with WebLogic Portal 9.2, see "Upgrading Autonomy Search Services" on page A-8.

CHAPTER 2

Upgrading to WebLogic Portal 9.2

This chapter describes upgrade tasks related to upgrading your WebLogic Portal *product* to version 9.2. Tasks related to upgrading individual *portal projects* that you created with previous releases of WebLogic Portal are described in the Upgrading WebLogic Portal Projects from Version 8.1 to Version 9.2 chapter in the *Portal Development Guide*.

Before You Begin

You can only upgrade WebLogic Portal version 8.1 SP4, SP5, and SP6 portal applications. For information on upgrading earlier versions to WebLogic Portal 8.1 SP4, SP5, or SP6, so that you can then upgrade to WebLogic Portal 9.2, see the Upgrade Guide for Version 8.1.

Database Changes During Upgrade

The WebLogic Upgrade Wizard executes database scripts to add and modify database tables for WebLogic Portal 9.2.

Note: Oracle 8.1.7 (DBMS version and drivers) is no longer supported. Upgrade to either Oracle 9i or 10G by following your vendor's instructions prior to upgrading to WebLogic 9.2.

The WebLogic Portal PointBase database is automatically upgraded when you run the WebLogic Upgrade Wizard. It is recommended that you keep the default option, which backs up the domain (including the existing `workshop.dbn` and `workshop$#.wal` files). During domain upgrade, the database files for `weblogic_eval` are renamed and other upgrade tasks are performed for the new version of PointBase, 5.1.

For additional information on database changes and how to perform database upgrade tasks manually, see Appendix B, "Performing Database Upgrade Tasks Manually."

Backing Up Your Applications and Data

BEA recommends that before upgrading your application environment, you manually back up the domain and any external application and application database resources in a separate process. You should back up the relevant information on all machines in the domain. The wizard backs up the domain directory only and does not preserve file permissions.

Upgrade Steps

This section provides general upgrade information for WebLogic Portal database and metadata files. The process consists of the following steps:

1. Verify that the WebLogic domain is not running.
2. Upgrade the portal domain using the WebLogic Upgrade Wizard before upgrading any applications.

   To start the WebLogic Upgrade Wizard in graphical mode and upgrade a WebLogic domain on a Windows platform, either:

   – Run `upgrade.cmd` from the *`WL_HOME`*`\common\bin` directory

   or

   – Go to **Start > All Programs > BEA Products > Tools > Domain Upgrade Wizard**

   Note: The WebLogic Upgrade Wizard is described in detail in *Upgrading WebLogic Application Environments*.

   WARNING: If you have a platform domain containing WebLogic Integration 8.1, you cannot perform a domain upgrade using this wizard. Contact Technical Support.

   The wizard upgrades your portal, content management, and personalization database data. It optionally upgrades your version 8.1 RDBMSAuthenticator to the version 9.2 WebLogic SQLAuthenticator. For more information on upgrading your user store, see "Security" on page 1-3.
3. Read Appendix A, "Functional Changes Affecting Your WebLogic Portal Environment".
4. As needed, upgrade individual applications, as described in the Upgrading WebLogic Portal Projects from Version 8.1 to Version 9.2 chapter of the Portal Development Guide.

Note: The *BEA Workshop for WebLogic Platform Programmer's Guide*, available as part of the Workshop for WebLogic help, contains several useful topics that you should review as you prepare to upgrade your portal application. The Workshop for WebLogic documentation includes step-by-step instructions for using the Import Wizard, and detailed information about what happens during the upgrade process and any required manual pre- or post-upgrade tasks.

Caution: You should be familiar with the Workshop for WebLogic upgrade steps, and any related limitations, before you attempt to upgrade a WebLogic Portal application from Version 8.1 4 or 8.1.5 to Version 9.2. Before proceeding, refer to the *Workshop for WebLogic* documentation on *e-docs* or by choosing **Help > Help Contents > *BEA Workshop for WebLogic Platform Programmer's Guide*** in the main menu of the product.

Review Functional Changes for WebLogic Portal 9.2

Review the functional changes that are described in Appendix A, "Functional Changes Affecting Your WebLogic Portal Environment." If any manual upgrade tasks are required for your particular environment, perform those tasks as instructed.

CHAPTER 3

Upgrading Domains to a Maintenance Pack

This chapter provides information about the domain upgrade tool, when to run the tool, how to run it, and finally how to test the results after running the tool to verify that the upgrade to a maintenance pack (MP) has been successful.

Before You Begin

Starting with Version 9.2 GA, shared J2EE libraries, known as library modules, are in use with WebLogic Integration (WLI), WebLogic Portal (WLP), and BEA Workshop for WebLogic Platform (WLW). Library modules are defined for inclusion in an enterprise or web application in a WebLogic Server `config.xml`.

The following is an example of a library module entry:

```
<library>
    <name>p13n-app-lib#9.2.0@9.2.0</name>
    <target>AdminServer</target>
    <source-path>[root]/common/deployable-libraries/p13n-app-lib.ear
    </source-path>
    <deployment-order>1</deployment-order>
    <security-dd-model>DDOnly</security-dd-model>
</library>
```

The <name> element includes the following components:

<name of the application or web library>

#

<specification versions>

@

<implementation version>

Implementation Versions

The procedure for upgrading domains from WebLogic Portal 9.2 to the available maintenance packs (9.2 MP1 and 9.2MP2) is the same. However, for each maintenance pack, the implementation version will be modified to reflect the new version of the library module. Thus, the `<name>` entry in the `<library>` element in a newly created domain in 9.2 MP2 will be: `<name>p13n-app-lib@9.2.0@9.2.2</name>`.

The implementation version for WebLogic Portal 9.2 MP2 is 9.2.2, and the implementation version for WebLogic Portal 9.2 MP1 is 9.2.1.

The domain upgrade tool provided with a maintenance pack will take the `config.xml` file of a previous 9.2 GA or MP installation and update the implementation versions for all the appropriate library modules of WLP and WLI domains. For 9.2 GA WLP and WLI domains, the upgrade tool also adds the required base patch libraries as `<library></library>` elements in the `config.xml` file. However, base patches are deployed automatically in WLP and WLI domains for all maintenance packs.

Steps to Run the Tool and Test the Upgrade

After you have installed the latest maintenance pack of WebLogic Portal, WebLogic Integration, or BEA Workshop for WebLogic Platform, the existing `config.xml` files will be pointing to invalid library modules of the previous installations. You must run the domain upgrade tool to update the `config.xml` files to point to the new library modules.

The domain upgrade tool is a set of scripts located in:`<weblogic_home>/common/bin`.

- UNIX scripts:
  - For WLW: `upgradeWLWConfigFile.sh`
  - For WLI: `upgradeWLIConfigFile.sh`

  - For WLP: `upgradeWLPConfigFile.sh`

- Windows scripts:
  - For WLW: `upgradeWLWConfigFile.cmd`
  - For WLI: `upgradeWLIConfigFile.cmd`
  - For WLP: `upgradeWLPConfigFile.cmd`

How to Run the Script

One parameter is required for running the above scripts: a fully qualified path to the `config.xml`.

Note: It is recommended to either make a backup copy of the `config.xml` file. The reason for this is that the changes are applied to the existing `config.xml`.

For example, to run the script for WLI, using a backup of the `config.xml` file in a temporary directory:

```
upgradeWLIConfigFile.cmd C:/tmp/config.xml
```

The script executes commEnv.cmd(on Windows) or commEnv.sh(on UNIX) in the `<weblogic_home>/common/bin` directory to set the environment for running the upgrade scripts.

The following message is displayed upon a successful upgrade of the `config.xml` file:

```
Your <config.xml_you_specified> file has been successfully upgraded.
```

For both WLI and WLP installations, this message will be displayed twice since the WLW library modules are upgraded first and then the appropriate WLI or WLP (or both) library modules are upgraded.

If you have patches applied on 9.2 GA, these entries are removed because the patches are rolled forward into the maintenance pack. Those patches and the entries in config.xml are not valid: the new library modules and library entries are effective after the upgrade.

Note: In a clustered environment, run the domain upgrade script only on the administrative service instance of the `config.xml`.

Testing the Domain Upgrade

After running the domain upgrade tool:

- The `config.xml` file will point to the latest maintenance pack versions of the shared library modules.

- Any 9.2 GA patch `<library>` entries added for a 9.2 GA patch will be removed.
- For WLI and WLP, base patch <library> entries will be added.

Take the upgraded `config.xml` file and copy it back into the domain's config directory (if a temporary location was used for the upgrade) and start the server via the `startWebLogic.cmd` or `startWebLogic.sh`.

Verify that no exceptions are thrown in the console and in the server log. Also verify, via WLS Console, that all application(s) are deployed and the implementation versions are correct for the maintenance pack you have installed.

Review Functional Changes for WebLogic Portal 9.2 and 9.2 MP1

Review the functional changes that are described in Appendix A, "Functional Changes Affecting Your WebLogic Portal Environment." If any manual upgrade tasks are required for your particular environment, perform those tasks as instructed.

Note: WebLogic Portal 9.2 MP2 does not contain any functional changes.

APPENDIX A

Functional Changes Affecting Your WebLogic Portal Environment

This appendix describes functional changes in WebLogic Portal Version 9.2 and Version 9.2 MP1 that affect your upgraded environment and might require you to perform manual tasks.

This chapter includes the following sections:

- Portlet State Persistence
- WSRP Security Compatibility
- Working with Encoding in HTTP Responses
- Disconnected Desktop Requires desktopStateShared Property
- Correcting Duplicate Portlet Category Names in an Upgraded Application
- Propagation Utility Web Application Obsolete

Functional Changes Introduced in WLP 9.2 MP1

The following changes have been introduced in WLP 9.2 MP1.

Domain Upgrade to Version 9.2 MP1

The domain upgrade tool provided in 9.2 MP1 will take an existing `config.xml` file and update the implementation version for all the appropriate product library modules. For information on when and how to run the tool, see "Upgrading Domains to a Maintenance Pack."

Enabling and Disabling Rich-text Editor

Rich text content in GroupSpace is susceptible to cross-site scripting (XSS) attacks. Because rich text content in GroupSpace is actually HTML, it is possible for users to add JavaScript code that will execute in other users' browsers when the HTML is rendered. It is possible that the JavaScript could contain malicious code, performing operations such as redirecting users' browsers. Examples of rich text content in GroupSpace include the body field in GroupNotes and the Notes field in Issues.

In WebLogic Portal 9.2 MP1, rich-text editing is controlled through the `web.xml` file by the administrator, and accordingly, the user interface for portal development displays options for rich-text and HTML editing.

Parameters in web.xml Related to the Rich-text Editor

The following use cases are affected by the `web.xml`.

- For GroupSpace Portlets (Issues, Announcements, GroupNotes):
  - If the `com.bea.apps.groupspace.RichTextEditorEnabled` parameter is set to "true" in `web.xml`, then the rich-text editor is made available based on the Portlet Preference, `RichTextEditorEnabled`. If the

`com.bea.apps.groupspace.RichTextEditorEnabled` parameter is not set, or is set to any value other than "true", the Portal Preference is not considered, and the rich-text editor is unavailable. If the value of this parameter is "true", then the Portlet Preference will determine if the editor is available.

  - When the rich-text editor is available, another parameter in `web.xml`, called `com.bea.apps.pgroupspace.RichTextEditorHTMLToolBarEnabled`, and a Portlet Preference, `RichTextEditorHTMLToolBarEnabled` control whether HTML can be entered directly or not. If the `web.xml` parameter is set to true, and the Portlet Preference is enabled, then the UI displays the HTML option, which allows entering HTML content in the rich-text editor.

- For collaboration portlets
  - If the `collaboration.RichTextEditorEnabled` parameter is to "true" in `web.xml`, then the rich-text editor is made available based on the Portlet Preference, `collaboration.rich_text.enabled`. If the parameter is not set in `web.xml`, or it is set to any other value than "true", then the Portlet Preference is not considered and the rich-text editor is unavailable. If the value of this parameter is "true", then the Portlet Preference will determine if the editor is available.

    Collaboration portlets do not support entering HTML directly in the rich-text editor.

Default Settings Related to the Rich-text Editor

This section describes the default settings related to the availability of the rich-text editor and HTML input.

Out of the box GroupSpace Sample

For out of the box GroupSpace sample, the rich-text editor will be enabled. For all other web projects, by default, the rich-text editor is not enabled for any portlet (GroupSpace or Collaboration). The administrator can change the settings, keeping in view the security risk.

A `web.xml` snippet will be included in that web application with the following settings:

```
com.bea.apps.groupspace.RichTextEditorEnabled = "true"

collaboration.RichTextEditorEnabled = "true"

com.bea.apps.groupspace.RichTextEditorHTMLToolBarEnabled = "true"
```

Note: Web projects created by customers in WebLogic Workshop will not contain any of the above parameters, and hence, the rich-text editor is disabled by default.

For GroupSpace (GroupNotes, Issues, Announcements, View Announcements), the Portlet Preference for `RichTextEditorEnabled` is “true” and for `RichTextEditorHTMLToolBarEnabled` is “false”.

User Projects

In the case of projects created by developers, the `web.xml` includes the following values:

```
com.bea.apps.groupspace.RichTextEditorEnabled = NOT SET
collaboration.RichTextEditorEnabled = NOT SET
com.bea.apps.groupspace.RichTextEditorHTMLToolBarEnabled = NOT SET
```

Note: NOT SET means “false”.

For GroupSpace, the Portlet Preference for `RichTextEditorEnabled` is “true” and for `RichTextEditorHTMLToolBarEnabled` is “false”.

For Collaboration Portlets, Portlet Preference for `collaboration.RichTextEditorEnabled` is “true” and HTML editing is not available.

Notes:

Modifying the parameter setting value in `web.xml` requires a redeployment of the web application in order to take effect.

Portal Preference values are cached in the pageflow. Therefore, after changing a preference value via the Admin Portal, you must log out of GroupSpace and log back in to see the effect of the new preference value.

Caution: These settings, in `web.xml` and at the Portlet Preference level, are not meant to be switched back-and-forth. You must decide at the time of deployment how you want the rich-text editor to be used and should establish these settings prior to using of the GroupSpace instance. If you create content with the rich-text editor, and then disable the editor, the display of the content can be illegible.

For example, when creating/editing content in the rich-text editor, new lines are represented as
 tags. However, when the rich-text editor is disabled, these will be displayed as “
” rather than causing a new line.

Enabling ResourceProxyServlet

The `ResourceProxyServlet` sends back to the browser Set-Cookie from static resources it retrieves. This Set-Cookie comes from the producer and will overwrite the Set-Cookie already in the browser which denotes the consumer session. Hence, the session will be lost when the next

request is received from the browser. Typically, setting the CookiePath would differentiate the Set-Cookies and solve the problem, but if that is not possible to do (for example, when you implement single sign-on in the same domain), the problem remains.

This problem has been resolved and a new system property is introduced, to prevent a sessions from being lost. Enabling the system property `wlp.resource.proxy.servlet.block.response.headers` blocks the `ResourceProxyServlet` from sending Set-Cookie header back to browser.

This allows you to keep the cookie names the same for both applications (as required for SSO) and still prevent the consumer's cookie from getting overwritten by the producer's cookie on the browser when a resource is returned.

For more information on setting the system property, see *Configuring Session Cookies* under http://edocs/wlp/docs92/federation/Chap-Best_Practices.html.

Functional Changes Introduced in WLP 9.2 GA

The following changes have been introduced in WLP 9.2 GA.

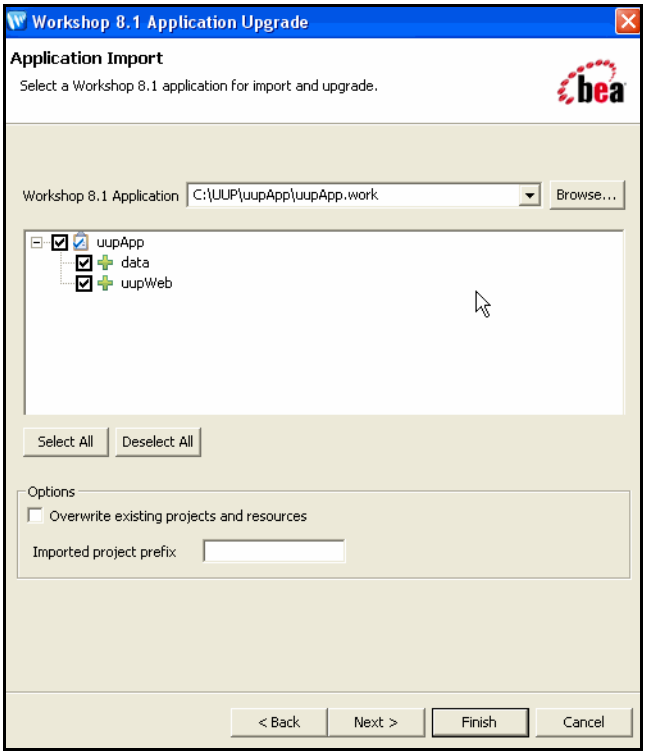
Upgrading UUPs

When you upgrade a UUP from WebLogic Portal 8.1, the `p13n_ejb.jar` file is deleted and replaced with a new WebLogic Portal 9.2 version of this file. The new `p13n_ejb.jar` file is packaged in the library modules that ship with WebLogic Portal 9.2.

Perform the following steps to upgrade a UUP configured in WebLogic Portal 8.1 to WebLogic Portal 9.2:

1. Start WebLogic Workshop and create a new Workspace.
2. Create a new portal domain. Do not create a Portal EAR Project. For instructions on creating a new domain, see the *Portal Development Guide*.
3. Import your Portal 8.1 UUP application into your new environment by choosing **File > Import**.
4. In the Select dialog, select **Workshop 8.1 Application** and click **Next**.
5. In the Application Import dialog, click **Browse** and locate your 8.1 UUP application. Select the `.work` file and click **Open**. Verify that the check boxes for the UUP application are selected and click **Next**, as shown in Figure A-1.

Figure A-1 Locate the 8.1 UUP Application



6. In the Source Upgrade dialog, select the **Use WebLogic 9.0 J2EE Shared Libraries** check box and you can also select the **Replace BEA NetUI tags with Apache Beehive tags** check box (if desired) and click **Finish**.

7. After the upgrade finishes, verify that the following actions occurred:

   – The `p13n-ejb.jar` file was removed from the EARContent directory of the UUP application.

   – The UUP EJB JAR file (for example, `UUPExample.jar`) exists in the EARContent directory of the UUP application.

   – The UUP EJB JAR file is referenced in a module entry in the `application.xml` file in the `<UUPApplication>/EARContent/META-INF/` directory.

   – As an example, the cache entry below was added to the `p13n-cache-config.xml` file in the `<UUPApplication>/EARContent/META-INF/` directory:

```
<p13n:cache>
     <p13n:name>UUPExampleCache</p13n:name>
     <p13n:description>Cache for UUP Example</p13n:description>
     <p13n:time-to-live>60000</p13n:time-to-live>
     <p13n:max-entries>100</p13n:max-entries>
</p13n:cache>
```

– Verify that the User Profile file (for example, `UUPExample.usr`) file exists in the `data/src/userprofiles/` directory (or where your Datasync folder exists).

8. Associate your portal application with your WebLogic Server by selecting the server in the **Servers** tab, right-clicking the server, and choosing **Add and Remove Projects**. Select the portal application from the **Available Projects** section, click **Add**, and then click **Finish**.
9. Build and publish your application. Verify the application by starting the WebLogic Server Administration Console and clicking **Deployments**. Verify that the UUP application is active. Then open the UUP application by expanding the tree and verifying that the UUP JAR file appears as an EJB.

Upgrading an 8.1 Java Project

If you had a WebLogic Portal 8.1 Java project and you upgrade to Portal 9.2, the Import Wizard automatically creates a Utility project. A Utility project is used to develop general-purpose Java code that is not directly part of special entities (web services, controls, or EJBs, for example). See Chapter 8 in the *Portal Development Guide* for instructions on creating a Utility project.

Table A -1 lists some of the common WebLogic Portal J2EE libraries that you might want to add to a Utility Project after you upgrade.

Table A -1 Common WebLogic Portal Libraries To Add To Utility Projects

J2EE Library	Purpose	Library Dependencies
`p13n-app-lib`	Base library required for all other WebLogic Portal libraries.	None.

J2EE Library	Purpose	Library Dependencies
wlp-services-app-lib	Contains portal application classes that are need to create custom placeholders.	p13n-app-lib
wlp-framework-full-app-lib	Contains the Portal Framework customization classes.	p13n-app-lib

See the *Portal Development Guide* for more information.

Upgrading Autonomy Search Services

If you are upgrading from WebLogic Portal 8.1 and used Autonomy search, you need to do the following to upgrade to the new version of Autonomy that is included with WebLogic Portal 9.2. The upgrade process does not automatically migrate your existing search settings and databases to the new search capabilities. You need to manually configure Autonomy search to work with your upgraded applications.

The new version of Autonomy is installed into the `portal/thirdparty/autonomy-wlp92` directory. For more information about configuring Autonomy search, see the *WebLogic Portal Search Guide*.

After installing and configuring Autonomy search, do the following to upgrade your application to use the new features:

1. Modify your WebLogic Portal 9.2 configuration files to match any Autonomy customizations you used.Table A-2 lists the files should be modified. If other configuration files were used that reference Autonomy search tools, you need to modify those as well.
2. Edit any start scripts that may start the 8.1 version of Autonomy services to ensure that they are not restarted. These are stopped automatically when you upgrade.

Table A-2 Mapping of WebLogic Portal 8.1 Configuration Files to WebLogic Portal 9.2 Configuration Files

File used with WebLogic Portal 8.1	File used with WebLogic Portal 9.2
PortalSearchDRE.cfg	AutonomyIDOLServer.cfg
PortalSearchDiSH.cfg	AutonomyDiSH.cfg

Table A-2 Mapping of WebLogic Portal 8.1 Configuration Files to WebLogic Portal 9.2 Configuration Files

File used with WebLogic Portal 8.1	File used with WebLogic Portal 9.2
`PortalSearchHTTPFetch.cfg`	`HTTPFetch.cfg`
`PortalSearchAutoIndexer.cfg`	`FileSystemFetch.cfg`

Using 8.1 Search within a 9.2 Portal Application

If you wrote applications using the WebLogic Portal `com.bea.query` classes or using the 8.1 Autonomy API and want to continue using these applications without using the new 9.2 version of the Autonomy API, you need to manually add the older, deprecated APIs to your application.

However, if you continue to use the 8.1 API (either WebLogic Portal or Autonomy's), those APIs will take priority over the 9.2 API and are NOT compatible with the 9.2 Autonomy engine. This means that some 9.2 content management features will not work, such as full-text search.

If you want to continue to use the 8.1 version of Autonomy with your portal application, you must manually add the older APIs to your installation.

To add the older Autonomy version to your 9.2 application:

1. Upgrade your domain and application to version 9.2.
2. Copy `autonomyCompat.jar` and `autonomyClient.1.5.0.jar` from `weblogic_home/portal/lib/thirdparty/search/81-compat-only` into the `application_home/APP-INF/lib` directory.
3. Using the Portal Administration Console, turn off full-text search for your repository. For more information, see the *WebLogic Portal Content Management Guide*.
4. Modify your domain start script to ensure that the 8.1 versions of the Autonomy are started rather than the new versions. For more information about start scripts, see the WebLogic Server documentation.
5. Start your domain.
6. Verify that search functionality is working.
   a. Index some content into Autonomy. For example, use FileSystemFetch.
   b. Execute a search in the Portal Search portlet.
   c. Verify that results are returned and no exceptions are encountered.

d. Add content to the BEA Repository instance using the content management tools.

e. Ensure that no exceptions are displayed; this would only occur if attempting to index the content, which will not occur if the preceding steps have been successfully executed.

The version 8.1 Autonomy APIs and engines are now running.

Upgrading to Use 9.2 Search after Using 8.1 Search with a 9.2 Portal Application

If you want to upgrade to use the 9.2 version of Autonomy, after you have completed the steps in "Using 8.1 Search within a 9.2 Portal Application" on page A-9, you can reverse the implementation.

To upgrade to use the 9.2 version of Autonomy (after you have implemented 8.1 Autonomy with 9.2):

1. Locate and update any usages of the `com.bea.query.*` classes or any calls to the Autonomy client APIs in your applications and replace them with the appropriate calls to the 9.2 version of the Autonomy API.
2. Remove the `autonomyCompat.jar` and `autonomyClient1.5.0.jar` file from the `app-inf/lib` directory.
3. Modify your domain start script to execute the 9.2 version of the Autonomy engines.
4. Configure the 9.2 Autonomy engines to index your content. For more information, see the *WebLogic Portal Search Guide*.
5. Using the Portal Administration Console, turn on full-text search for your repository. For more information, see the *WebLogic Portal Content Management Guide*.
6. Run the startup script.

Manually Upgrading Passwords in Content Management Repositories

After the upgrade is complete, you must manually re-enter the passwords for your third-party repositories using the Administration Console; see the *Content Management Guide* for more information about editing repository settings.

Until you manually re-enter the passwords for your third-party repositories, you cannot access those repositories.

Maintaining Content Queries

In WebLogic 8.1 through WebLogic Portal 8.1 SP5, content query expressions were generated differently due to an order of precedence problem. The order of precedence was not maintained when executing a content query expression. For example, the following expression: (a && (b || c), gets evaluated/executed as (a && b || c).

This problem was fixed in 9.2 such that the order of precedence is now maintained when executing a content query expression. However, if you want to continue using the WebLogic Portal 8.1 through WebLogic Portal SP5 query behavior, you need to modify your domain scripts to define the following system property: `-Dwlp.disable.content.rule.fix=true`.

Upgrading Look & Feels

Portal Look & Feels in WebLogic Portal 8.1 used two configuration files for skins and skeletons (in the `/skins/<skin_name>` and `/skeletons/<skeleton_name>` directories): `skin.properties` and `skeleton.properties`. Both were text files, and `skeleton.properties` was optional.

In WebLogic Portal 9.2, both files are now XML, and both are required.

To upgrade a WebLogic Portal 8.1 Look & Feel to the WebLogic Portal 9.2 format:

1. Make sure the portal application containing the Look & Feel has been converted to WebLogic Portal 9.2, as described in the Portal Development Guide.
2. Open the Look & Feel in Workshop for WebLogic and re-save it. The configuration files are automatically converted to the new XML format.

Import Wizard Does Not Handle cm_taglib.jar

The `cm_taglib.jar` is the tag library for the WebLogic Portal 7.0-based DocumentManager feature. The Import Wizard will not detect whether or not you have any JSPs that refer to this taglib, which has an unsupported taglib URI. The JSPs will fail.

The `cm_taglib.jar` file was not installed by default in a new 8.1 web application, but if you added it to your application for backward compatibility, you must handle this file manually in your upgraded application.

Change all references to the `cm_taglib.jar` so that they use supported tags and APIs, and delete the file `cm_taglib.jar`.

Changes in Behavior Between Struts 1.1 and 1.2

WebLogic Portal support for Struts is slightly different in Version 9.2 if you upgrade to Struts 1.2.

Struts 1.1 support in WebLogic Portal is the same as in previous releases, with our struts-adapter taglibs mapped to URIs using `web.xml`. You can use the `struts-1.1.war` library module instead of the new `struts-1.2.war` library module.

If you are upgrading to Struts 1.2, instead of mapping the struts and struts-adapter taglibs using `web.xml`, WebLogic Portal now relies on the JSP 1.2 implicit taglib mapping, wherein any `.tld` files in the `META-INF` directory in a JAR are implicitly mapped by the web container to the URI specified in the tld. For WebLogic Portal. these are in `struts-adapter.jar`, in the path `META-INF/tlds`.

Choose to use one of these two methods to upgrade to Struts 1.2:

- Modify all JSPs that use the struts taglibs to reference `http://bea.com/struts/adapter/tags-html` and `http://bea.com/struts/adapter/tags-nested` for the HTML and nested taglibs, and `http://struts.apache.org/tags-*` for the remainder of the taglibs that our adapter does not override.
- Extract the `.tld`s from both `struts.jar` (in `struts-1.1.war`) and from `struts-adapter.jar` (in our portal web library module) and copy them to `WEB-INF/tlds`. This allows for the case where you want to continue using the explicit tld mapping via `web.xml`.

Definition Labels Not Editable in Version 9.2

In Version 8.1 the capability to edit definition labels existed, but was not recommended. Modifying the definition label could have unintended implications; for example, exposing a protected resource or breaking WSRP (which uses the definition label as the portlet handle).

In Version 9.2 this functionality has been replaced by a much richer ability to move portal resources (books, pages, desktops) between production and development environments, without losing user customizations or changing labels. These new features include XIP and the propagation utility. XIP allows you to target individual portal resources to import and export between development and production systems, and you can specify the scope (library, admin, or visitor). For more information about WebLogic Portal's propagation tools, see the Production Operations Guide.

Portlet State Persistence

In WebLogic Portal 8.1, minimized portlet states were persisted only for the session. You can use a workaround, described in the *Upgrade Guide for Version 8.1*, to set up a backing file that controls the states of portlets under the desktop.

The solution used in Version 8.1 will continue to work if you depend on this behavior, but BEA recommends that you use a new method of persisting the portlet state.

In Version 9.2, you can persist the portlet state in the database by using the `persistence-enabled` attribute in the `control-state-location` element of the `netuix-config.xml` file.

Here is an example of the `persistence-enabled` attribute:

```
<control-state-location>
    <session persistence-enabled="true"/>
</control-state-location>
```

If you set the `persistence-enabled` attribute to `true`, then the control tree state will be loaded from the database when a user logs in, and the new state will be stored when a user logs out. The control tree state is cleared when a user interacts with a portal.

The default and the only persistence type implemented is JDBC. The default implementation uses the BEA_PORTAL_FRAMEWORK_CONTROL_TREE_STATE property set of the user's ProfileWrapper; the ProfileWrapper must be created and stored in the user's http session. The ProfileWrapper is created by PortalServletFilter, which should be configured in the `web.xml` file. If the ProfileWrapper is not found in the user's session, the control tree state is not persisted.

Note: Page flow- and struts-related states are not persisted, as they are not part of the control tree state; page flow and struts portlets might not be in the exact same state when user logs out.

The child elements reader-class-name/writer-class-name provide reader and writer class names for this state-location. If you want to customize reader/writer behavior, you can implement the ControlStateReader/ControlStateWriter interfaces and configure them in the `netuix-config.xml` file.

WSRP Security Compatibility

Producer and consumer applications developed with WebLogic Portal 9.2 are compatible with producers and consumers developed with WebLogic Portal 8.1. That is, a portal developed with

WebLogic Portal 9.2 can consume portlets deployed in a WebLogic Portal 8.1 domain. Similarly, portlets exposed in a WebLogic Portal 9.2 producer can be consumed by an 8.1 consumer.

However, if you want to use your own key for the 8.1 or 9.2 consumer, you need to follow the procedures outlined the chapter "Establishing WSRP Security with SAML" in the *Federated Portals Guide*.

Working with Encoding in HTTP Responses

This section describes changes in how the encoding is set on the HTTP response.

Version 8.1 Methodology in Setting Encoding

In Version 8.1 of WebLogic Portal, the following method of setting encoding was used:

1. Examine the `.portal` file for a `directive.page` element. If that element is present, obtain the encoding from an attribute there.
2. If the element is not present, use the JSP encoding configuration, which looks at the `<encoding>` element in the `<jsp-param>` section of the `web.xml` file; the default is ISO-8859-1 if these elements are missing.

Both of these mechanisms are now deprecated.

Version 9.2 Methodology in Setting Encoding

In Version 9.2 of WebLogic Portal, the following method of setting encoding is used:

1. Examine the `netuix:desktop` element for an `encoding` attribute, and use that value if present.
2. If the first check is not applicable, examine the `.portal` file for the deprecated `directive.page` element. If that element is present, pick up the encoding from an attribute there.
3. Examine `netuix-config.xml` for a `<defaultEncoding>` element, and use the `encoding` attribute there.
4. If the previous check is not applicable, fall back to the deprecated `<encoding>` element in the `<jsp-param>` section of the `web.xml` file.

The old methods continue to work, but to eliminate any deprecation warnings, BEA recommends that you use the new mechanisms; for example:

```
<netuix:desktop ... encoding="UTF-8" /> in your .portal file
```

or

```
<defaultEncoding encoding="UTF-8" /> in your netuix-config.xml file
```

Editing Encoding in Workshop for WebLogic

When you select the desktop element from the portal view or outline view while editing a `.portal` file, the workbench now includes a new `encoding` property in the property view. The value for a new portal defaults to `UTF-8`. You can edit the value using an editable drop-down menu.

The drop-down menu initially displays a list of all the basic IANA encoding values as well as the extended encodings for Chinese, Korean, and Japanese. The values presented in the list are descriptive display names that are converted to actual IANA names when saved to the `.portal` file.

If a desired value does not display in the drop-down menu, you can type it in. When you press Enter, the validator checks the encoding to verify that it is a valid and supported encoding. The value you enter can be from the extended encoding set and can be an IANA name, alias, or canonical name for the encoding. If the encoding fails validation, a warning message displays; you can choose to override the validation and accept the value anyway. The value will be stored as is, in the `.portal` file for the desktop `encoding` attribute.

Disconnected Desktop Requires desktopStateShared Property

For backward compatibility and to support the few desirable uses of a disconnected desktop, WebLogic Portal Version 9.2 provides a new, but deprecated, boolean property called `desktopStateShared` for the `StandalonePortletURL` and the associated JSP tag. You can use this property to retain the previous “disconnected desktop” behavior. The default value of this property and attribute will be `true`, causing the portlet to be connected to a desktop.

Explicitly setting either the `path` or `contextualPath` properties on the URL or tag also disassociates the resulting standalone portlet from its originating desktop. This also holds true for any URLs generated within the context of the standlone portlet—setting `path` or `contextualPath` causes the resulting URLs to become disassociated with the originating desktop.

Correcting Duplicate Portlet Category Names in an Upgraded Application

In past releases of WebLogic Portal it was possible, though not recommended, to create more than one portlet category with the same name, at the same level in the hierarchy. In Version 9.2 this operation is not permitted. (You can use the same name for more than one category, but they must not be "peers" in the hierarchy.)

When you upgrade a portal application to Version 9.2, any duplicate portlet category names that were used previously are preserved. It is extremely important that you edit these category names to be unique; otherwise the WebLogic Portal propagation tools might cause unexpected results, or errors might occur during the propagation process.

Propagation Utility Web Application Obsolete

The Propagation Utility web application, `propagation.war`, is obsolete in WebLogic Portal 9.2. This application that was introduced in a patch for WebLogic Portal 8.1 SP4 and later incorporated into WebLogic Portal 8.1 SP5. If you are upgrading a WebLogic Portal web application in which you previously installed the file `propagation.war` into the root directory of any WebLogic Portal enterprise applications, it is recommended that you remove the file before or after upgrading to WebLogic Portal 9.2.

APPENDIX B

Performing Database Upgrade Tasks Manually

This appendix describes how to perform various database upgrade tasks manually if you do not use the WebLogic Upgrade Wizard to perform upgrade from 8.1 SP4, SP5, or SP6.

Files associated with database upgrade are located in the following directories:

WL_HOME\portal\upgrade\db

WL_HOME\common\p13n\upgrade\db

This appendix contains the following sections:

PointBase Name Changes

PointBase stores all data in `.dbn` files and all log information in `.wal` files. Database properties are stored in `PointBase.ini` files. Data files for WebLogic Portal are now named `weblogic_eval.dbn` and log files for WebLogic Portal are named `weblogic_eval$1.wal`. They were previously named `workshop.dbn` and `workshop$1.wal`, respectively.

Upgrading your Main WebLogic Portal Database

If you did not upgrade your main WebLogic Portal database using the WebLogic Upgrade Wizard, you can perform the upgrade manually as follows:

1. Shut down WebLogic Server.
2. Back up your database data as described by your database vendor.
3. Edit the *`WL_HOME`*`\portal\upgrade\db\upgrade_db.properties` file for your database environment. Replace the @ symbols and the text between the symbols with the correct values for `@DB_USER@`, `@DB_PASSWORD@`, `@DB_HOST@`, `@DB_PORT@`, and `@DB_NAME@`.

   Note: For PointBase, follow the instructions in "Upgrading PointBase Databases" on page B-3.

4. Set `files=` as follows (this is the default):

   ```
   files=${env.WL_HOME}/common/p13n/db/${database}/jdbc_index/UPGRADE/jdbc
   .index,${env.WL_HOME}/portal/db/${database}/jdbc_index/UPGRADE/jdbc
   .index
   ```

   Note: If you are using SQL Server, and you applied the patch for CR244320 to your SQL Server database, *do not* use the above `files=` setting; instead set `files=` as follows:

   ```
   files=${env.WL_HOME}/common/p13n/db/${database}/jdbc_index/UPGRADE/
   unicode/jdbc.index,${env.WL_HOME}/portal/db/${database}/jdbc_index/
   UPGRADE/unicode/jdbc.index
   ```

5. Either:

   – Run `upgrade_db.cmd/.sh`

   or

   – Set `noexec=Y` and `SQLOutputFile=outputfile.sql`, and manually run the output SQL file generated with this `files=` setting.

6. Run `upgrade_db_data.cmd/.sh` to upgrade database data.

Tip: To upgrade your user store, follow the instructions in "Upgrading to the WebLogic Server SQL Authenticator" on page B-4.

7. Determine if patch # CR244936 has been applied. If this patch has been applied, manual upgrade of the main WebLogic Portal database is complete. This patch is described in BEA WebLogic Portal 8.1 Service Pack 5 Release Notes (`http://edocs.beasys.com/wlp/docs81/relnotes/relnotes.html#1117746`) as follows:

   CR237251 - If markup contains more than 4000 bytes, an attempt to store it in the database causes an error. If your database does not contain a PF_MARKUP_XML table that has been populated with data i.e. Select count(*) from PF_MARKUP_XML returns 0 rows) and the PF_MARKUP_DEFINITION table does not contain the BEGIN_XML and END_XML columns then this patch was not applied.

8. If patch # CR244936 has not been applied, run the script:

   WL_HOME`\portal\db\`*dbms_name*`\pf9_drop_columns.sql`

Upgrading PointBase Databases

To upgrade a PointBase database manually:

1. Shut down WebLogic Server and the PointBase Server.
2. Copy the `workshop.dbn` and `workshop$#.wal` files from the 8.1 domain directory to the upgrade directory.
3. Rename the `workshop.dbn` file to `weblogic_eval.dbn`. Rename the `workshop$#.wal` file `weblogic_eval$#.wal`.
4. Run `upgrade_db.cmd/.sh` to process required WebLogic Portal database updates for PointBase.
5. Run `upgrade_db_data.cmd/.sh` to upgrade database data.
6. Copy the upgraded `weblogic_eval.dbn` and `weblogic_eval$#.wal` files back to the domain directory. The Upgrade Wizard will have updated the URL to reflect the database name change and will have modified the `pointbase.ini` file for the 5.1 version of PointBase used with WebLogic Portal 9.2.

Upgrading to the WebLogic Server SQL Authenticator

If you did not upgrade your user store using the WebLogic Upgrade Wizard during the domain upgrade process, you can perform a manual upgrade later. The script to upgrade from the WebLogic Portal-specific RDBMS Authenticator to the WebLogic SQL Authenticator is:

WL_HOME\common\p13n\db*dbms_name*\upgrade_fromdbmsauth_tosqlauth.sql

Upgrading Separate Behavior Tracking Databases

If you created a separate behavior tracking database in version 8.1, upgrade it as follows:

1. Shut down WebLogic Server.
2. Back up your database data as described by your database vendor.
3. Edit the settings in the `upgrade_db.properties` for your behavior tracking database. Replace the @ symbols and the text between the symbols with the correct values for `@DB_USER@`, `@DB_PASSWORD@`, `@DB_HOST@`, `@DB_PORT@`, and `@DB_NAME@`.
4. Modify the `files=` setting in the `upgrade_db.properties` file and set it to:

   ```
   files=${env.WL_HOME}/portal/db/${SQL_DIR}/jdbc_index/UPGRADE/BT/jdbc
   .index
   ```

5. Either:
   - Run `upgrade_db.cmd/.sh`

   or

   - Set `noexec=Y` and `SQLOutputFile=outputfile.sql`, and manually run the output SQL file generated with this `files=` setting.

Upgrading Additional Content Management Databases

The default content management database is upgraded automatically. If you created an additional content management database in 8.1, upgrade it as follows:

1. Shut down WebLogic Server.
2. Back up your database data as described by your database vendor.

3. Update the settings in the `upgrade_db.properties` for your content management database. Replace the @ symbols and the text between the symbols with the correct values for `@DB_USER@`, `@DB_PASSWORD@`, `@DB_HOST@`, `@DB_PORT@`, and `@DB_NAME@`.
4. Modify the `files=` setting in the `upgrade_db.properties` file and set it to:

   ```
   files=${env.WL_HOME}/portal/db/${SQL_DIR}/jdbc_index/UPGRADE/CM/jdbc
   .index
   ```

   Note: If you are using SQL Server and have already applied the patch for CR244320 to your SQL Server database, instead modify the `files=` setting in the `upgrade_db.properties` file to:

   ```
   files=${env.WL_HOME}/portal/db/${SQL_DIR}/jdbc_index/UPGRADE/CM/unicode
   /jdbc.index
   ```

5. Either:
   - Run `upgrade_db.cmd/.sh`

   or

   - Set `noexec=Y` and `SQLOutputFile=outputfile.sql`, and manually run the output SQL file generated with this `files=` setting.

Dropping Deprecated RDBMS Authenticator Tables After Upgrade

After you have upgraded to the WebLogic Server SQL Authenticator, you can drop the tables associated with the WebLogic Portal RDBMS Authenticator using the following script:

WL_HOME\common\p13n\db*dbms_name*\dep9_drop_tables.sql

Dropping Deprecated Compoze Database Tables After Upgrade

After you have upgraded to version 9.2, you can drop the tables associated with Compoze/Collaboration using the following script:

WL_HOME\portal\db*dbms_name*\dep9_drop_tables.sql