BEA WebLogic Portal®

Upgrade Guide

Version 8.1 Service Pack 6
Revised: July 2006

Copyright

Copyright © 2004-2006 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, BEA WebLogic Server, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic JRockit, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server Process Edition, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

Contents

About This Document

What You Need to Know xiv
The e-docs Web Site. xiv
How to Print the Document xv
Related Information xv
Contact Us xv
Documentation Conventions xvi

1. Overview of the Upgrade Process from WebLogic Portal 7.0 SP2 to 8.1

Definitions 1-1
Portal Version 7.0 SP2 to 8.1 Upgrade Process Overview. 1-2
Comparing Supported Features 1-2
Look and Feel 1-3
Stylesheets. 1-3
Properties 1-4
Scripts 1-4
Images. 1-4
Skeletons. 1-4
JSP 1-4
Layouts 1-4
Layout Definition 1-5
Group Portals. 1-5

Webflow Support and Limitations 1-5
Exceptions 1-6
Page Flows 1-6
Portal Services 1-6
Security 1-6
Commerce and Personalization 1-6
Struts 1-7
How to Upgrade Existing Applications 1-7
Compatibility Domain 1-7
Upgrading to Weblogic Portal 8.1 1-8
Conversion of Portal and Portlet Definition Files 1-8
Portal Web Applications 1-8
Look and Feel Components 1-9

2. Version 8.1 Compatibility Domain

Hosting 7.0 Applications in 8.1 Compatibility Domain 2-1
Compatibility Explained 2-1
Features and Limitations of the Compatibility Domain 2-2
Development Issues 2-2
XA - Transaction Level Support 2-2
MBean Configuration Mechanism 2-2
Runtime and Administrative Issues 2-3
Listing Parent Groups 2-3
Authentication Providers in Portal Compatibility Domain 2-3
User and Group Management 2-3
Procedure for Hosting in Compatibility 2-4
Creating a Portal Compatibility Domain 2-5
Before You Begin 2-5
Step 1: Create New Portal Domain 2-5

Step 2: Edit Start Script 2-6
Step 3: Configure the New Domain 2-7
Step 4: Configure the Database 2-7
Upgrade Existing Data 2-10
Security Upgrade 2-10
Install and Configure the RDBMS Authentication Provider 2-11
Upgrading the Enterprise Application 2-13
Before You Begin 2-13
Procedure for Each Enterprise Application 2-14
Content Management Configuration 2-19
Modify Code 2-24
Upgrade Application-Sync Files 2-28
Final Adjustments to the Domain 2-28
Add Users to PortalSystemAdministrators Group 2-28
Upgrade CustomerRole in SSPI to Point to RDBMS Groups 2-29
Users and Groups Specified in fileRealm.properties 2-30

3. Upgrading to WebLogic Portal 8.1

Upgrading from Compatibility Mode to WebLogic Portal 8.1 3-1
Before You Begin 3-1
Procedure for Upgrading from Compatibility 3-2
Adding Catalog Administration 3-4
Group Portals 3-5
Struts Support 3-5
Struts Applications in WebLogic Portal 7.0 3-5
Support for Struts in WebLogic Portal 8.1 3-5
Upgrading Struts Applications to Page Flows in WebLogic Portal 8.1 3-6
JSP Tag Replacements 3-6
Upgrading Database and Metadata Files from 7.0 to 8.1 3-6

Step 1: Upgrade PointBase Triggers 3-7
Step 2: Upgrade WebLogic Portal 7.0 Schema 3-8
Step 3: Upgrade Application-Sync Files 3-9
Step 4: Upgrade Portal and Portlet Files 3-9
Step 5: Upgrade ENTITLEMENT_RULESET Records 3-13
Step 6: Upgrading Existing Behavior Tracking Data. 3-14

4. Upgrading 8.1 to Service Pack 2

Step 1: Upgrade an Existing Domain or Create a New SP2 Domain 4-1
Upgrading an Existing Domain 4-2
Creating a New SP2 Domain 4-4
Step 2: Upgrade Existing Applications and Projects. 4-4
Before You Begin – About UUP 4-4
Update the Portal Libraries 4-4
Step 3: Redeploy the Upgraded Application. 4-5

5. Upgrading to Service Pack 3

The Service Pack Upgrade Process. 5-1
Step 1: Upgrade an Existing SP2 Domain or Create A New SP3 Domain. 5-2
Upgrading an Existing SP2 Domain 5-2
WebLogic Portal Upgrade Installer 5-2
WebLogic Portal Full Installer 5-2
Creating a New SP3 Domain 5-4
Step 2: Upgrade Existing Database Schema 5-5
Upgrading a PointBase Database. 5-5
Remove OLD_ Tables After Upgrade (Optional). 5-6
Upgrading a Sybase Database 5-6
Remove OLD_ Tables After Upgrade (Optional). 5-8
Upgrading a SQL Server Database 5-8

Remove OLD_ Tables After Upgrade (Optional) 5-9
Upgrading an Oracle or DB2 Database 5-9
Remove OLD_ Tables After Upgrade (Optional) 5-9
Step 3: Upgrade Existing Applications 5-10
Re-Encrypting Password 5-10
Encrypting Passwords 5-11
Note on Changing Passwords 5-12
Before You Begin – About UUP 5-13
Update the Portal Libraries 5-13
Step 4: Redeploy the Upgraded Application 5-14

6. Upgrading to Service Pack 4

The Service Pack Upgrade Process 6-2
Step 1: Upgrade an Existing SP2 or SP3 Domain or Create A New SP4 Domain 6-2
Upgrading an Existing SP2 or SP3 domain 6-2
WebLogic Portal Upgrade Installer 6-3
WebLogic Portal Full Installer 6-4
Creating a New SP4 Domain 6-6
Step 2: Upgrade Existing Database Schema 6-6
Upgrading a PointBase Database 6-6
Upgrading a Sybase Database 6-8
Upgrading a SQL Server Database 6-10
Upgrading an Oracle or DB2 Database 6-11
Step 3: Upgrade Existing Applications 6-12
Re-Configure Third-Party Content Management Systems 6-12
Re-Encrypting Passwords for WSRP Producers 6-13
Encrypting Passwords 6-14
Note on Changing Passwords 6-16
Updating Portal Libraries 6-16

Before You Begin – About UUP. 6-16
Update the Portal Libraries 6-16
Step 4: Redeploy the Upgraded Application 6-18
Step 5: Review Functional Changes for SP4. 6-18

7. Upgrading to Service Pack 5

The Service Pack Upgrade Process 7-1
Step 1: Upgrade an Existing SP3 or SP4 Domain or Create A New SP5 Domain 7-2
Upgrading an Existing SP3 or SP4 Domain 7-2
WebLogic Portal Upgrade Installer 7-2
WebLogic Portal Full Installer 7-3
Creating a New SP5 Domain 7-5
Step 2: Upgrade Existing Portal Framework Data 7-5
Step 3: Upgrade Existing Applications 7-6
Re-Configure Third-Party Content Management Systems 7-6
Re-Encrypting Passwords for WSRP Producers 7-7
Encrypting Passwords 7-8
Note on Changing Passwords 7-10
Updating Portal Libraries. 7-10
Before You Begin – About UUP. 7-10
Update the Portal Libraries 7-10
Step 4: Redeploy the Upgraded Application. 7-12
Step 5: Review Functional Changes for SP5. 7-12

8. Upgrading to Service Pack 6

The Service Pack Upgrade Process 8-1
Step 1: Upgrade an Existing SP4 or SP5 Domain or Create A New SP6 Domain 8-2
Upgrading an Existing SP4 or SP5 Domain 8-2
WebLogic Portal Upgrade Installer 8-2
WebLogic Portal Full Installer 8-3

Creating a New SP6 Domain 8-5
Step 2: Upgrade Existing Applications 8-5
Re-Configure Third-Party Content Management Systems. 8-5
Re-Encrypting Passwords for WSRP Producers 8-6
Encrypting Passwords 8-7
Note on Changing Passwords 8-9
Updating Portal Libraries 8-9
Before You Begin – About UUP 8-9
Update the Portal Libraries 8-10
Step 3: Redeploy the Upgraded Application 8-11
Step 4: Review Functional Changes for SP6 8-11

A. Functional Changes Affecting Your WebLogic Portal Environment

Portlet State Persistence (Base Release, SP5) A-2
New Option for Request Attribute Persistence (SP4) A-3
New Generic URL Tag (SP6) A-4
Oracle 10g R2 Database Requires Manual Privilege Setup A-5
Associating Portlets with Pages (Base Release, SP5) A-5
Mandatory Portlets (8.1 Base Release) A-6
Upgrading Simple Producers from Service Pack 3 . . . . . A-6
Aggressive Control Tree Persistence (SP4) A-7
Content Management Portlet (SP4) A-7
URL Template File Updates (SP4) A-7
Creating New Portals Using Administration Portal A-7
URL Template File Update for WSRP A-8
Updates for Java Portlets A-8
Portlet Preferences Behavior on Server Restart (SP4) A-8
Entitlements Data Source (SP4) A-9
Storage of Policy Reference Data in RDBMS (SP4) A-9

Using Role Caching with Entitlements (SP4) A-10
Setting up JDBC Connection Polling (SP4) A-10
Portal Projects in Source Control (SP4) A-10
Using New Proxy Portlet Cache (SP6) A-11
Using Content Query Expressions (SP6) A-12
Propagating Request Parameters from WSRP Requests (SP6) A-12

B. Changes to Visitor Tools for SP4

Functional Changes B-1
Code Changes B-1
Internationalization B-1
Moving JavaScript to .js Files B-2
Changes to JSP Files B-2
Changes to Java Files B-2
Unchanged Files B-4

C. Database Changes for SP4

Entitlement Policy Reference C-1
Portal Framework C-2
Sybase Database Changes C-2
SQL Server Database Changes C-3
DB2 Database Changes C-3

D. Database Changes for SP3

Virtual Content Management D-1
Web Services for Remote Portlets (WSRP) D-2
WebLogic Commerce Services (CMV) D-2
Portal Framework D-3

E. Framework Reference for Portal Upgrades from 7.0 to 8.1

Security E-1
Security Framework JSPs. E-1
Authentication Providers E-1
Unified User Profile (UUP) E-2
Connecting to an Active Directory Server E-3
Portal Properties. E-3
Portal Component Properties E-3
Desktop Properties E-6
Header and Footer Properties E-6
Book and Page Properties E-7
Placeholder Properties. E-9
Portlet Type Properties E-10
Portlet Instance Properties. E-13
Framework File Reference E-14
Framework Files. E-14
JSP Reference E-16
JSP Tag Changes E-16
JSP Tag Libraries E-17
Notes on Individual JSP Tags E-19

About This Document

This document includes procedures for upgrading WebLogic Portal applications built on Version 7.0 SP2 to run on Version 8.1; it also describes upgrading from earlier Version 8.1 service packs to Service Pack 5.

This document covers the following topics:

- Chapter 1, "Overview of the Upgrade Process from WebLogic Portal 7.0 SP2 to 8.1," gives an overview of the procedures and elements to be considered when undertaking any upgrade project.
- Chapter 2, "Version 8.1 Compatibility Domain," describes the features and limitations of the Portal Compatibility Domain, and details the process for creating one and for upgrading an existing 7.0 SP2 application to run within it.
- Chapter 3, "Upgrading to WebLogic Portal 8.1," details significant changes to the WebLogic Portal platform architecture, including implementation considerations where possible.
- Chapter 4, "Upgrading 8.1 to Service Pack 2," provides instructions for applying service pack changes to your portal applications after you install the service packs. It walks you through the steps of creating a new SP2 domain, upgrading existing applications and projects, and redeploying the upgraded application.
- Chapter 5, "Upgrading to Service Pack 3," provides instructions for applying service pack changes to your portal applications after you install the service packs. It walks you through the steps of creating a new SP3 domain, upgrading the existing Version 8.1 database schema, upgrading existing applications and projects, and redeploying the upgraded application.

- Chapter 6, "Upgrading to Service Pack 4," provides instructions for applying service pack changes to your portal applications after you install the service packs. It walks you through the steps of creating a new SP4 domain, upgrading the existing Version 8.1 database schema, upgrading existing applications and projects, and redeploying the upgraded application.
- Chapter 7, "Upgrading to Service Pack 5," provides instructions for applying service pack changes to your portal applications after you install the service packs. It walks you through the steps of creating a new SP4 domain, upgrading the existing database markup, upgrading existing applications and projects, and redeploying the upgraded application.
- Appendix A, "Functional Changes Affecting Your WebLogic Portal Environment," describes functional changes in WebLogic Portal Version 8.1 and service packs that affect your upgraded environment and might require that you perform manual tasks.
- Appendix B, "Changes to Visitor Tools for SP4," describes the functional and general code changes made to the Visitor Tools for SP4.
- Appendix C, "Database Changes for SP4," includes details about WebLogic Portal Version 8.1 database changes for SP4.
- Appendix D, "Database Changes for SP3," includes details about WebLogic Portal Version 8.1 database changes for SP3.
- Appendix E, "Framework Reference for Portal Upgrades from 7.0 to 8.1,", includes details about the WebLogic Portal Framework and the changes between Version 7.0 and 8.1.

What You Need to Know

This document is intended mainly for application developers and architects needing to upgrade an existing WebLogic Portal application in the new platform. It assumes a familiarity with the WebLogic Portal platform and Java programming.

The e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click Product Documentation or go directly to the "e-docs" Product Documentation page at http://e-docs.bea.com.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File → Print option on your Web browser.

A PDF version of this document is available on the WebLogic Portal documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Portal documentation Home page, click the PDF files button and select the document that you want to print.

If you do not have the Adobe Acrobat Reader, you can obtain it for free from the Adobe Web site at http://www.adobe.com/.

Related Information

The following BEA WebLogic Portal documents contain information that is relevant to using the idltojava compiler and understanding how to implement Java CORBA applications in the WLE system.

For more information in general about Java IDL and Java CORBA applications, refer to the following sources:

- The OMG Web Site at http://www.omg.org/
- The Sun Microsystems, Inc. Java site at http://java.sun.com/

Contact Us

Your feedback on the BEA WebLogic Portal documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Portal documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Portal 8.1 release.

If you have any questions about this version of BEA WebLogic Portal, or if you have problems installing and running BEA WebLogic Portal, contact BEA Customer Support through BEA WebSupport at www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`

Convention	Item
monospace italic text	Identifies variables in code. *Example*: `String expr`
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`...`	Indicates one of the following in a command line: • That an argument can be repeated several times in a command line • That the statement omits additional optional arguments • That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER 1

Overview of the Upgrade Process from WebLogic Portal 7.0 SP2 to 8.1

This section provides an overview of the strategies and procedures for upgrading WebLogic Portal 7.0 SP2 applications to WebLogic Portal 8.1 and its associated service packs. The following topics are discussed:

- Definitions
- Comparing Supported Features
- How to Upgrade Existing Applications

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, 7.0 to 8.1.

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 7.0 SP2 to 8.1 Upgrade Process Overview

Two paths are available for upgrading your WebLogic Portal 7.0 SP2 portal web application to WebLogic Portal 8.1:

- **Compatibility:** Allows for the configuration of an existing WebLogic Portal 7.0 SP2 portal web application to run on the 8.1 portal server.

  Hosting an existing Portal Web application in a Portal Compatibility Domain requires following the procedure outlined in "Hosting 7.0 Applications in 8.1 Compatibility Domain" on page 2-1.

- **Upgrade:** Allows for the upgrade of WebLogic Portal 7.0 SP2 portal applications and resources to WebLogic Portal 8.1.

  Hosting an existing Portal Web application in a WebLogic Portal 8.1 domain requires following the procedure outlined in "Upgrading from Compatibility Mode to WebLogic Portal 8.1" on page 3-1.

Comparing Supported Features

This section outlines significant feature changes between the WebLogic Portal 7.0 SP2 and the WebLogic 8.1 release. For detailed descriptions and implementation details, consult Appendix E, "Framework Reference for Portal Upgrades from 7.0 to 8.1."

The portal framework in WebLogic Portal 7.0 has a one-to-one mapping with the portal it represents; this methodology adheres to the single portal per web application architecture. Changes made to this framework will impact the portal and all group portals, with limited provisions for changes based on runtime factors. This code was not originally designed to be modified by end users and was instead considered part of the product itself. However, users who wished to change the way that the portal behaved or make significant changes to the way that the portal looked were forced to modify this framework.

In WebLogic Portal 8.1, the framework has been designed to allow users to more readily modify the look and the feel of the product. Instead of the single collection of JSP files for a portal, any portal can use a look and feel, which in turn is comprised of a skin and a skeleton. The portal

framework is an XML skeleton that replaces the framework JSPs, allowing users to easily modify the behavior of the portal without modifying the underlying BEA code.

Look and Feel

The look and feel of a portal is determined primarily by the code used to render the HTML, the styles used by the HTML, and the images displayed. In 7.0 only the skin could be changed to obtain a different look and feel, but in 8.1 there is a look and feel document that sets the skin and the skeleton, with the latter being the framework that is used to render the elements.

The look and feel file used in 8.1 is an XML document with an `.laf` extension that defines the name, the skin to use, and the skeleton to use. This allows one skeleton to be used with multiple skins, or vice-versa, with the former being the most likely case. Creating or modifying the look and feel file in 8.1 involves working with XML directly as there is no visual editor provided.

Stylesheets

WebLogic Portal 8.1 uses Cascading Stylesheets (CSS) to a greater extent than 7.0, with more tags and improved structure.

In WebLogic Portal 7.0, a single `main.css` file contains entries for portal, portlet, pages, and other elements using a single-level naming convention. Examples include `.titlebar`, `.portletcontainer`, `.pageheader`, and so on. Creating a new skin includes modifying the main.css file and setting the desired values for these tags.

In WebLogic Portal 8.1, multiple files are used for this task:

- `body.css`: Portal body (header, footer, and so on) styles
- `book.css`: Book, page, and menu styles
- `button.css`: Button styles
- `fix.css`: Browser bug-fix styles
- `form.css`: Form, input, and text area styles
- `layout.css`: Layout and placeholder styles
- `window.css`: Portlet styles

The naming convention in WebLogic Portal 8.1 is multi-level, providing more granularity and flexibility when using styles. Examples include bea-portal-book-primary-menu,

bea-portal-window-titlebarcontainer, bea-portal-body-footer, and so on. As with WebLogic Portal 7.0, creating a new skin includes copying an existing skin and modifying the styles.

Properties

In WebLogic Portal 8.1, a `skin.properties` file contains entries that define the paths to images, links, scripts, and so on. In most cases it will not be necessary to modify this file, but the option to do so provides increased flexibility for locating resources.

Scripts

There are a few JavaScript functions that are used in WebLogic Portal 8.1 for popup menus, initialization, and so on. These can be modified on a per-skin basis, but it is recommended that application specific scripts be placed in separate files.

Images

Skins in the new release use fewer application-specific images. These images can be modified when creating custom skins.

Skeletons

The skeletons used in WebLogic Portal 8.1 are roughly equivalent to the framework JSPs in WebLogic Portal 7.0, but the code, tags, and overall structure are quite different. There is no direct path to transfer WebLogic Portal 7.0 framework modifications to skeleton in WebLogic Portal 8.1, but a JSP developer should be able to do the conversion manually.

JSP

The JSPs in a skeleton directory use scriptlets and JSP tags to render the various elements of the portal. Files include `body.jsp`, `book.jsp`, `popupmenu.jsp`, and so on, which correspond to the portal elements. Modifications to these files can change the way that the elements are rendered as well as the way that they behave.

Layouts

Layouts are similar to those in WebLogic Portal 7.0, but have added flexibility and functionality. The layouts are derived from three base types: grid, border, and flow. These format types place elements on a grid, at the center, north, south, east, or west, or in a horizontal or vertical flow, respectively.

Layout Definition

The layouts in WebLogic Portal 8.1 are defined in XML in the form of `.layout` files. These files specify the type of layout to use (grid/border/flow), the number of columns, the HTML to use for the tools, and naming. When creating a new layout it is easiest to start with an existing layout definition and modify it.

- HTML Files

  The HTML files included in layouts in WebLogic Portal 8.1 are for use by WebLogic Workshop and the WebLogic Administration Portal. These HTML files use specific classes to specify the layout and placeholders, and are similar to the JSPs. The HTML is used to render the layouts and modify the contents of placeholders.

- JSP Files

  The JSP files included in layouts in WebLogic Portal 8.1 are used to render the layout on a page and are part of the skeleton. There is one file for each layout type, including `gridlayout.jsp`, `borderlayout.jsp`, and `flowlayout.jsp`. You can modify these layout types as part of the skeleton, but in general it is suggested that custom layouts be created if these do not behave in the desired way. For example, a spanning layout can allow portlets to span rows or columns.

Group Portals

Group portal definitions are not upgraded: in WebLogic Portal 8.1, the corresponding mechanism is the Desktop, by which a bundle of portal components is presented to a specified audience.

Webflow Support and Limitations

Portlet Webflow files can be upgraded, and can be used in conjunction with Page Flows.

WebLogic Workshop allows Webflow support to be added to existing WebLogic Portal 8.1 applications. This means Webflow portlets can run inside a WebLogic Portal 8.1. All Webflow presentation node types can be supported. An entire Webflow file can be supported, along with all its semantics, such as chaining.

Webflow and Page Flow portlets can be used in a common page. Page Flows can call Webflow files, allowing existing pipeline components and input processors to be leveraged. Page Flows can call out to Webflows and map the return to a forward in the Page Flow.

Exceptions

Page Flow JSPs cannot call validators. Webflow presentation nodes can be supported as long as they are the endpoint of a Webflow.

Page Flows

Provided they call the Webflow's entry point (using the syntax origin - namespace - event) Page Flows can call Webflow presentation nodes.

The resulting WebflowResponse can be used to construct a Java Page Flow Forward using the URI or URL constructors, but the processor nodes are definitely still in the loop.

Portal Services

Content from an existing WebLogic Portal 7.0 repository can be consumed and leveraged into any existing integration or process. The WebLogic Portal 8.1 Virtual Content Repository allows the content management administration tools to be used against content in a WebLogic Portal 7.0 repository. Additionally, a new bulkloader tool supports moving content from the flat WebLogic Portal 7.0 repository into WebLogic Portal 8.1 repositories, enabling you to take advantage of the new content management features.

Security

WebLogic Portal 8.1 implements the WebLogic Server SSPI and uses the default LDAP store for user information.

- Existing customers will be able to continue to leverage their RDBMS user stores by leveraging an SSPI authentication provider that will be provided for connecting to a RDBMS.
- Entitlement segments on WebLogic Portal 7.0 cannot be upgraded because of the change to leveraging the WLS SSPI. Customers will need to create roles using the WebLogic Administration Portal for entitlements.

Commerce and Personalization

The only significant change to commerce functionality is that the JSP tools for orders, payments and catalog are no longer included in the WebLogic Administration Portal.

Note: For instructions on adding Catalog Administration tools to an out-of-the-box WebLogic Portal 8.1 application, consult the "Adding Catalog Administration" section in "Procedure for Upgrading from Compatibility" on page 3-2.

Regarding personalization, property set schema and values remain unchanged and require no upgrade.

Behavior tracking requires current data be archived and re-hosted in a new schema. The procedure is outlined in "Step 6: Upgrading Existing Behavior Tracking Data" on page 3-14.

Struts

Some customers might have existing Struts-based applications or desire to start developing Struts applications on WebLogic Portal 7.0 because of the move to Page Flows in the WebLogic Portal 8.1 release, which are built on Struts 1.1. Guidelines on hosting Struts-based applications in WebLogic Portal 8.1 are listed in "Struts Support" on page 3-5.

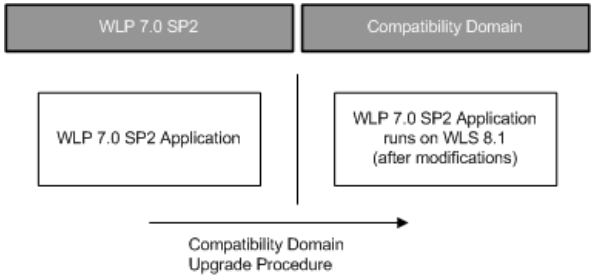
How to Upgrade Existing Applications

When considering the route to run your WebLogic Portal 7.0 SP2 applications on WebLogic Portal 8.1, two choices are available immediately: to host the application to run in a Portal Compatibility Domain, as explained in "Hosting 7.0 Applications in 8.1 Compatibility Domain" on page 2-1, or to upgrade the application to run in WebLogic Portal 8.1, as explained in "Upgrading from Compatibility Mode to WebLogic Portal 8.1" on page 3-1. The bulk of this guide is devoted to the Portal Compatibility Domain path, but also includes significant information required for re-implementation with WebLogic Portal 8.1.

Compatibility Domain

An existing Weblogic Portal 7.0 application can be converted to run in a Portal Compatibility Domain using the procedure detailed in "Hosting 7.0 Applications in 8.1 Compatibility Domain" on page 2-1.

Figure 1-1 Upgrade to Compatibility Domain



Upgrading to Weblogic Portal 8.1

Listing , "Upgrading from Compatibility Mode to WebLogic Portal 8.1," on page 3-1 includes the information required for re-implementing an existing Weblogic Portal 7.0 application to run in WebLogic Portal 8.1. The capabilities and limitations of this configuration are explained here, along with a general description of new platform features included in WebLogic Portal 8.1.

Figure 1-2 Re-Implementation for WebLogic Portal 8.1

Conversion of Portal and Portlet Definition Files

WebLogic Portal 8.1 includes the command line tool `upradePortalFiles.cmd` (`.sh`) that converts `.portal` and `.portlet` files from version 7.0 SP2 to version 8.1. This file resides in the *`WL_HOME`*`\portal\upgrade` directory. For more information, see "Step 4: Upgrade Portal and Portlet Files" on page 3-9.

Portal Web Applications

Portal Web applications built on WebLogic Portal 7.0 SP2 can be run on WebLogic Portal 8.1, using the upgrade procedure contained in "Upgrading from Compatibility Mode to WebLogic Portal 8.1" on page 3-1.

Look and Feel Components

WebLogic Portal 7.0 skins and UI components can be re-hosted to Look and Feels and appropriate components in the WebLogic Portal Version 8.1. For details on Look and Feel implementation, see "Connecting to an Active Directory Server" on page E-3.

CHAPTER 2

Version 8.1 Compatibility Domain

Hosting 7.0 Applications in 8.1 Compatibility Domain

WebLogic Portal 8.1, through Service Pack 2, supports a Compatibility Domain to enable upgraded WebLogic Portal 7.0 portal applications to be hosted on the newer version of the product. This section includes instructions on creating a Compatibility Domain, as well as steps required to upgrade your existing portal application to run in such a domain.

Note: The WebLogic Portal 8.1 Compatibility Domain originally supported applications built on WebLogic Portal 7.0 SP2. The WebLogic Portal 8.1 SP2 Compatibility Domain supports applications built on WebLogic Portal 7.0 SP4. The WebLogic Portal 8.1 Compatibility Domain is supported only through Service Pack 2.

Compatibility Explained

The 8.1 release of WebLogic Platform introduces significant changes to the platform in which Portal applications run. To ease the transition from WebLogic Portal 7.0 to 8.1, the Compatibility Domain enables 7.0 portal applications to be run on the new platform with minimal alterations.

The Compatibility Domain is a feature introduced in WebLogic Portal 8.1 in which an existing 7.x portal (and domain), possibly in a cluster, with existing data in the database, can be 'upgraded' to run in a specially-configured Compatibility Domain. This allows you to take advantage of the new platform features in new applications while also running existing applications within the same instance of WebLogic Platform.

Features and Limitations of the Compatibility Domain

This section explains features and limitations in a Compatibility Domain for the purpose of upgrade planning and strategy.

Generally speaking, portals built in WebLogic Portal 7.0 which are re-hosted to run in a Portal Compatibility Domain can be administered using the Weblogic 8.1 Administration Portal. No WebLogic Portal 8.1 features are supported in a portal enterprise application running in a Portal Compatibility Domain. Development tools such as the E-Business Control Center, WebLogic Workshop (any version) are not supported against applications running in a Portal Compatibility Domain.

Development Issues

This section includes some considerations that might affect your upgrade planning, depending on how your applications are implemented.

XA - Transaction Level Support

In the Compatibility Domain, the `com.bea.p13n.util.jdbc.SequencerFactory` is not backward compatible. In order to support XA, the sequencer must have its own datasource. So all the APIs where you could pass one in have been removed. The "Framework File Reference" on page E-14 includes information meant to assist in refactoring existing code. You can also consult the Javadoc at the following link:

`http://edocs.bea.com/workshop/docs81/doc/en/portal/buildportals/navReference.html`

MBean Configuration Mechanism

In WebLogic Portal 7.0, the portal MBeans (mail service, campaign service, ad service, behavior tracking, events, caches, etc.) were configured in the WebLogic Server console. A portal plug-in for the server console allowed you to change these settings.

In Portal Compatibility Mode, this portal plug-in is not present. In WebLogic Portal 8.1, these settings are exposed in the Portal Administration Tools, which do not run against an application in a Portal Compatibility Domain.

The solution is to edit the `application-config.xml` file (in the `APP/META-INF` directory), change the settings, then re-start the server.

Runtime and Administrative Issues

This section explains issues concerning running and administering applications in a Portal Compatibility Domain, and discusses differences between WebLogic Portal 7.0 and 8.1 applications.

Portal Compatibility Domain supports all runtime features of WebLogic Portal 7.0 with the following exceptions:

- In the WebLogic Portal Administration Tools, Payment Management will not be available.
- FileRealm users and groups will not be automatically available.

The Compatibility Domain supports a slightly limited security model: WebLogic Portal 7.0 portals work with users and groups in both the RDBMS and LDAP realms (with a single provider) and those in `fileRealm.properties`. A WebLogic 8.1 Portal Domain supports users in a single provider (RDBMS or LDAP). The net difference is that the fileRealm users and groups will no longer be automatically available in a Portal Compatibility Domain. These are typically system users and groups.

Listing Parent Groups

In Portal Compatibility Domain, the WebLogic Administration Portal does not list parent groups to which a user belongs, a change from WebLogic Portal 7.0. If groups A and B were subgroups of C, and a user were a member of A and B, the User Details screen would list all three groups in WebLogic Portal 7.0 Administration Portal, but only groups A and B in the Portal Compatibility Domain Administration Portal.

Authentication Providers in Portal Compatibility Domain

This section explains authentication considerations that might arise when creating new portals meant to run inside a Portal Compatibility Domain.

User and Group Management

Due to the design of WebLogic Portal 8.1, Portal User and Group Management can work with a single Security Authentication Provider (the first one listed in the console). This is a domain-level configuration—all Portal enterprise applications (more specifically, User and Group Management calls by the portal applications) in the domain can see only users and groups in the first Security Authentication Provider.

Default WebLogic Portal 8.1 domains are configured to use the default authentication provider (LDAP). By default, portals use LDAP to store and access users and groups.

A Portal Compatibility Domain uses the `RDBMSAuthenticationProvider` to provide access to the users and groups from the WebLogic 7.0 SP2 domain, because generally these were stored in the database. Therefore, portals running in a Portal Compatibility Domain will generally use RDBMS to store and access users and groups.

If the out-of-the-box WebLogic Portal 8.1 domain and the Portal Compatibility Domain use different authentication providers, there are two choices:

- Configure the domain to use RDBMS Authentication, and manually migrate LDAP users and groups (from the out-of-the-box WebLogic Portal 8.1 domain) to the RDBMS. Passwords are not preserved in this procedure.
- Configure the domain to use LDAP Authentication, and manually migrate RDBMS users/groups (from the 'upgraded' portal) to LDAP. Passwords are not preserved in this procedure.

Note: If the WebLogic 7.0 SP2 domain uses LDAP instead of RDBMS authentication for users and groups, the Portal Compatibility domain will likewise use LDAP, meaning it will use the same provider as the out-of-the-box WebLogic Portal 8.1 domain.

To avoid this issue, any new portals designed in an out-of-the-box WebLogic Portal 8.1 domain should be configured to use `RDBMSAuthenticationProvider`.

This method requires installing the `RDBMSAuthenticationProvider` and designating it as the first provider in the list in the console, directly following the creation of the new domain.

Procedure for Hosting in Compatibility

The process for hosting an existing WebLogic Portal 7.0 application to run inside a Portal Compatibility Domain includes three phases:

1. Creating a Portal Compatibility Domain
2. Upgrading the Enterprise Application
3. Final Adjustments to the Domain

Note: This procedure is based on the following assumptions:

The compatibility domain is called `portalCompatibilityDomain` and is created at `user_projects\domains\portalCompatibilityDomain`.

The “upgraded to compatibility” portal applications directory is `user_projects\applications\...` (the apps can be stored anywhere)

Creating a Portal Compatibility Domain

This section explains the steps necessary to create a Portal Compatibility Domain.

Before You Begin

This procedure requires that the following elements be in place:

- Complete WebLogic Portal 8.1 installation
- Existing WebLogic Portal 7.0 domain
- `db_settings.properties` file from the existing domain

Portal Compatibility starts with the portal domain configured using the portal Configuration Wizard template, and continues with a manual process for configuring the domain and 7x enterprise applications. A primary goal is that very few code changes will be required for a `domain/ent-app` that uses only Portal. Most of the configuration steps involve replacing `.jar` files and editing configuration files.

Discover SystemAdministrator Users from Existing Domain

Start the WebLogic Portal 7.0 server, open the Portal Administration Tools, and make a list of the users in the SystemAdministrator group. You will need this when upgrading the domain.

Back Up Existing Domain

Make backup copies of the existing domain, enterprise applications, and of course, the database.

Step 1: Create New Portal Domain

Use the Configuration Wizard in WebLogic Portal 8.1 to create a new Portal domain, using the Express setup option and the template for the Basic WebLogic Portal Domain. Use the same domain username and password as you used for the existing WebLogic Portal 7.0 domain.

Note: This procedure illustrates a domain called “`portalCompatibilityDomain`”, but it can be named anything.

1. Copy the `dmsBase` directory from the WebLogic Portal 7.0 domain directory to the Portal Compatibility Domain directory.

2. Transfer commerce properties.

   a. If you were using commerce features in the WebLogic Portal 7.0 domain, copy the `weblogiccommerce.properties` file from the `weblogic700\portal\` directory to the `weblogic81\portal\` directory.

   b. Next, modify the compatibility domain's `startWeblogic` script to reference this newly copied properties file.

      For example, add the following line to the script:

      ```
      -Dcommerce.properties=<absolute path to weblogiccommerce.properties>
      ```

Step 2: Edit Start Script

Make the following edits to the `startWeblogic.cmd` script for the new WebLogic Portal 8.1 Compatibility Domain:

- For all databases:

  Add this line just before the server startup line (roughly 330 - beginning with "if) `%WLS_REDIRECT_LOG%"==""  ('):`

  ```
  set
  CLASSPATH=%WLP_HOME%\lib\portal_system.jar;%WLP_HOME%\lib\commerce_syst
  em.jar;%CLASSPATH%
  ```

- For PointBase only:

  Configure the database host, port, instance name, and the WebLogic Server host and port, in the following scripts, using the values in your `db_settings.properties` file as a guideline:

  – `startWebLogic.cmd/.sh`

    For example, change this:

    ```
    call "%WL_HOME%\common\bin\stopPointBase.cmd" -port=9093
    -name=workshop ...
    ```

    to this, assuming you used the default PointBase instance and port:

    ```
    call "%WL_HOME%\common\bin\stopPointBase.cmd" -port=9092
    -name=wlportal ...
    ```

  – `stopWebLogic.cmd/.sh`

    For example, change this:

```
call "%WL_HOME%\common\bin\startPointBase.cmd" -port=9093 ...
. . .
call "%WL_HOME%\common\bin\stopPointBase.cmd" -port=9093
-name=workshop ...
```

to this, assuming you used the default PointBase instance and port:

```
call "%WL_HOME%\common\bin\startPointBase.cmd" -port=9092 ...
. . .
call "%WL_HOME%\common\bin\stopPointBase.cmd" -port=9092
-name=wlportal ...
```

Step 3: Configure the New Domain

Edit the following settings for the new domain:

- If the existing WebLogic Portal 7.0 domain had SSL enabled, enable SSL for the new WebLogic Portal 8.1 Compatibility Domain by editing `config.xml` and setting `Enabled="true"` in the `<SSL>` entry under the `<Server>` section.
- Server port settings (Listen Port) are in the `config.xml` file in the existing WebLogic Portal 7.0 domain directory. Verify that they are the same in the `config.xml` file for the new domain.

Step 4: Configure the Database

The upgrade process requires that the new domain have access to the existing database. Database settings are contained in files within the domain directory for both 7.0 and 8.1. The following steps ensure that the database settings in the new 8.1 domain match those in the existing 7.0 domain.

1. Set the `JDBCConnectionPool` entry.

   Configure the `<JDBCConnectionPool>` URL settings in the 8.1 Portal Compatibility Domain's `config.xml` file to correspond to the URL settings in the `config.xml` file for the existing WebLogic Portal 7.0 domain, matching the database host, port, and schema name. Listings 2-1 and 2-2 show sample `JDBCConnectionPool` entries for a new WebLogic Portal 8.1 Compatibility Domain.

Listing 2-1 JDBCConnectionPool Entry for PointBase

```
<JDBCConnectionPool Name="cgPool" Targets="portalServer"
CapacityIncrement="1"
DriverName="com.pointbase.jdbc.jdbcUniversalDriver"
```

```
InitialCapacity="5" MaxCapacity="50"
Password="{3DES}dYceZKN2mwcfajtJC6uzSg=="
Properties="user=weblogic;" RefreshMinutes="0"
ShrinkPeriodMinutes="15" ShrinkingEnabled="true"
SupportsLocalTransaction="true" TestConnectionsOnRelease="false"
TestConnectionsOnReserve="false"
URL="jdbc:pointbase:server://localhost:9092/wlportal"/>
<JDBCDataSource JNDIName="weblogic.jdbc.pool.commercePool"
Name="commercePool" PoolName="cgPool" Targets="portalServer"/>
```

Listing 2-2 JDBCConnectionPool Entry for Oracle

```
<JDBCConnectionPool CapacityIncrement="1"
DriverName="oracle.jdbc.driver.OracleDriver"
InitialCapacity="25" MaxCapacity="25" Name="cgPool"
Password="weblogic"
Properties="user=weblogic"
RefreshMinutes="0" ShrinkingEnabled="false"
Targets="portalServer" TestConnectionsOnReserve="false"
URL="jdbc:oracle:thin:@<dbhost>:1521:<dbSID>"/>
<JDBCDataSource JNDIName="weblogic.jdbc.pool.commercePool"
Name="commercePool" PoolName="cgPool" Targets="portalServer"/>
```

Note: A JDBC datasource has been added to Listing 2-2 for access to the WebLogic Portal 7.0 data.

2. Update the database configuration files.

   Update Portal Compatibility Domain files to reference correct database configuration settings:

   Configure the 8.1 Portal Compatibility Domain's `db_settings.properties` file to correspond to the settings in the `db_settings.properties` file for the existing WebLogic Portal 7.0 domain, matching the host, db_name, port, dblogin, dbpassword, and connection fields. Listings 2-3 and 2-4 show sample configuration settings for a new WebLogic Portal 8.1 Compatibility Domain.

Listing 2-3 Database Configuration Entry for PointBase

```
#@IF_USING_POINTBASE@
database=POINTBASE
db_version=44
jdbcdriver=com.pointbase.jdbc.jdbcUniversalDriver
host=localhost
db_name=portal
port=9093
dblogin=WEBLOGIC
dbpassword=WEBLOGIC
connection=jdbc:pointbase:server://localhost:9093/wlportal
pointbase_ini=pointbase/pointbase.ini
#@ENDIF_USING_POINTBASE@
```

Listing 2-4 Database Configuration Settings for Oracle

```
database=ORACLE_THIN
db_version=817
jdbcdriver=oracle.jdbc.driver.OracleDriver
server=<dbSID>
port=1521
dblogin=USERNAMEGOESHERE
dbpassword=PASSWORDGOESHERE
connection=jdbc:oracle:thin:<dbHost>:1521:<dbSID>
```

3. Prepare PointBase files for the schema upgrade.

   The PointBase instance requires a few preparatory steps before the schema upgrade:

   Follow the steps in "Step 1: Upgrade PointBase Triggers" on page 3-7. These steps prepare the PointBase files in `upgrade/pointbase` to be schema-upgraded.

4. Upgrade the database schema.

Before the actual data can be migrated from the existing database, the schema must be updated. This applies to all databases, which require the preparatory steps described in "Step 2: Upgrade WebLogic Portal 7.0 Schema" on page 3-8.

Upgrade Existing Data

This step updates the contents of the `ENTITLEMENT_RULESET` table.

All Databases

Follow the steps in "Step 5: Upgrade ENTITLEMENT_RULESET Records" on page 3-13.

Specifically, the following changes occur in this procedure:

For each row in the `ENTITLEMENT_RULESET` table, this modifies the value of the `RULESET_DOCUMENT` column as follows: The XML text `http://www.bea.com/servers/p13n/xsd/rules/core/2.1.1 rules-core-2_1_1.xsd` is prepended to the `<cr:rule-set>` element's `xsi:schemaLocation` attribute value.

PointBase Only

For PointBase only, copy the upgraded `.wal` and `.dbn` files from the `upgrade/pointbase/` directory to the WebLogic Portal 8.1 Compatibility Domain directory (or wherever your `.wal/.dbn` files are).

Security Upgrade

The remainder of the steps required to prepare the new Portal Compatibility Domain involve upgrading security. The following tasks are described in this section:

- Install an `RDBMSAuthenticationProvider` to access existing users and groups stored in the RDBMS.
- Add to the RDBMS any portal users and groups previously in `fileRealm.properties` required for access for the portal running in portal compatibility, and reset their passwords.
- Manually add the PortalSystemAdministrators group to the RDBMS; it is required by portal compatibility (and WebLogic Portal 8.1 in general) to make changes in the WebLogic Portal Administration Tools.
- Add users with portal administrator privileges in the existing WebLogic Portal 7.0 domain to the `PortalSystemAdministrators` group.

- Reconfigure any ACLs that were configured in the WebLogic Portal 7.0 `fileRealm.properties` (or on an LDAP store); they will not be automatically migrated by the upgrade process.

Install and Configure the RDBMS Authentication Provider

The RDBMS authentication provider allows users and groups stored in a WebLogic Portal 7.0-based RDBMS store to be accessed from a portal in release 8.1 using the WebLogic Security Service Provider Interface (SSPI). This allows the WebLogic 8.1 Portal to work with users and groups defined in WebLogic Portal 7.0's RDBMSRealm, with minimal changes.

You can skip Step1 and Step2 below if your 7.0 domain did not use the RDBMSRealm. If you do not know whether you used the RDBMSRealm, open the `config.xml` file in the WebLogic Portal 7.0 domain. If the file does not contain a stanza for `<RDBMSRealm>`, then your 7.0 domain did not use the RDBMSRealm and you can skip to "Step 3: Add PortalSystemAdministrators Group Using Outside Authentication Store" on page 2-13. If `config.xml` does contain this stanza, check the value of BasicRealm in the `<CachingRealm>` stanza; if it matches the `Name` in the `<RDBMSRealm>`, then your 7.0 domain did use the RDBMSRealm and you must complete Step1 and Step 2 in this section.

Step 1: Configure the Authentication Provider

Use this procedure to configure the authentication provider in the new Compatibility Domain:

1. In the new WebLogic Portal 8.1 Compatibility Domain, run the startWeblogic script and navigate to the following URL in the browser: http://localhost:7501/console. You might need to adjust the host name and port to match the settings in `config.xml`.
2. Login with a user ID of `weblogic` and a password of `weblogic`. Choose **Security→Realms→myrealm→Providers** and select **Authentication**.
3. Click **Configure a new RDBMSAuthenticator**, then set the control flag to `SUFFICIENT`, and click **Create**.
4. Click the Details tab and edit the database settings according to the `config.xml` and `db_settings.properties` in the previous section.

   For example, using `sampleportal` with PointBase, the settings would be as shown in Listing 2-5; using `sampleportal` with Oracle, the settings would be as shown in Listing 2-6.
5. Click **Apply.**
6. Choose **Security→Realms→myrealm→Providers** and select **Authentication**.

7. Select `DefaultAuthenticator` (the LDAP provider), set the control flag to SUFFICIENT, and click **Apply**.
8. Select **Security→Realms→myrealm→Providers** and select **Authentication**.
9. Click **Re-order the Configured Authentication Providers**, make the `RDBMSAuthenticator` appear first in the list, and click **Apply**.
10. Shut down the server.

Listing 2-5 Authentication Settings for PointBase

```
Database Driver: com.pointbase.jdbc.jdbcUniversalDriver
Database URL: jdbc:pointbase:server://localhost:9092/wlportal
Schema Properties:
user=WEBLOGIC;password=WEBLOGIC;server=jdbc:pointbase:server://localhost:9
092/wlportal
Minimum Password Length: whatever it was in 7.x (ie 1 character for 7x
sampleportal)
```

Listing 2-6 Authentication Settings for Oracle

```
Database Driver: oracle.jdbc.driver.OracleDriver
Database URL: jdbc:oracle:thin:@myOraDB:1521:myOraInstance
Database Username: weblogic
Database Password: weblogic
Schema Properties: user=weblogic;password=weblogic
Minimum Password Length: 8
```

Step 2: Add PortalSystemAdministrators Group Using RDBMSAuthentication

This section is applicable only if your 7.0 domain used the RDBMSRealm.

In order to support using the WebLogic Portal 7.0 Administration Tools, you must add the PortalSystemAdministrators group to the `RDBMSAuthenticationProvider`. If users and groups were stored in the database in the existing WebLogic Portal 7.0 domain, you must add the PortalSystemAdministrators group to RDBMSAuthentication.

Using the SQL editor of your choice (for example, SQL*Plus or PointBase Console), run the following SQL:

```
insert into GROUP_SECURITY( GROUP_ID, GROUP_NAME ) values ( 900,
'PortalSystemAdministrators' );

insert into ENTITY( ENTITY_ID, ENTITY_NAME, ENTITY_TYPE ) values ( 900,
'PortalSystemAdministrators', 'Group' );
```

Note: 900 is a reserved number in this index, so there should be no problem with this operation.

Step 3: Add PortalSystemAdministrators Group Using Outside Authentication Store

This step is applicable only if your 7.0 domain *did not* use the RDBMSRealm.

If users and groups were not stored in the database in the existing WebLogic Portal 7.0 domain, add the PortalSystemAdministrators group to the LDAP (or other) store.

You might want to migrate any ACLs in the `fileRealm.properties` file of the existing WebLogic Portal 7.0 domain.

Upgrading the Enterprise Application

This section describes how to configure a WebLogic Portal 7.0 enterprise application to run in the WebLogic Portal 8.1 Compatibility Domain that you created by following the instructions in the Creating a Portal Compatibility Domain section.

In the next section, `ENT-APP` refers to the directory for your enterprise application, and `WEB-APP` refers to the directory for your web application(s).

Before You Begin

This section includes instructions for re-hosting enterprise applications to run in the new WebLogic Portal 8.1 Compatibility Domain. Make sure you have performed the backup steps described in the Before You Begin section of "Creating a Portal Compatibility Domain" on page 2-5.

For sampleportal only:

1. Make sure the values in the `db_settings.properties` file in your 7.0 domain directory are consistent with those in your 8.1 Compatibility Domain. If you have changed your 8.1 settings to match the 7.0 settings, you can skip the remaining steps in this section.

2. Run `configca.bat/.sh` from your 7.0 domain directory. This creates an updated version of `BlackBoxNoTx.rar` in your 7.0 enterprise application directory.
3. Copy `BlackBoxNoTx.rar` from your enterprise application directory to the `customerservice` portlet directory. For example:

```
cp\bea70\weblogic700\samples\portal\sampleportalDomain\beaApps\
sampleportal\BlackBoxNoTx.rar\bea70\weblogic700\samples\portal\
sampleportalDomain\beaApps\sampleportal\sampleportal\portlets\
customerservice
```

Procedure for Each Enterprise Application

For each enterprise application to be upgraded for the new domain, follow these steps:

1. Copy the enterprise application directory from the existing WebLogic Portal 7.0 domain to the WebLogic Portal 8.1 Compatibility Domain.
2. Delete the following objects:
   - **/Datasync** subdirectory
   - **/tools** subdirectory
   - **/toolSupport** subdirectory
   - All BEA-provided enterprise-level `.jar` files, `.war` files, `.ear` files, and `.rar` files for the enterprise application (that is, those files in the enterprise application directory). For example, if you were upgrading sampleportal, you would delete all .jar, .war, and .rar files from the `<8.1 domain>\applications\sampleportal` directory. Remove *only* BEA-provided files.
3. Create Enterprise Application Directory with the following hierarchy: `<ENT-APP dir name>\APP-INF\lib`.

   For example, if you were upgrading sampleportal you would create *`WL_HOME`*`\samples\portal\applications\sampleportal\APP-INF\lib`.
4. Copy the following files from the `<bea81>\weblogic81` directory to your new `APP-INF\lib` directory
   - `portal\lib\commerce\ejb\commerce_util.jar`
   - `portal\lib\netuix\system\ext\ejb\dom4j-full.jar`
   - `portal\lib\netuix\ejb\netuix_util.jar`

- portal\lib\wps\ejb\wps_util.jar
- portal\lib\portal\ejb\portal_util.jar

5. Copy the following files from the <bea81>\weblogic81 directory to your <*ENT-APP dir_name*> directory:

- portal\lib\content.jar
- portal\lib\content_repo.jar
- portal\lib\tools700.war
- portal\lib\toolSupport.war
- portal\lib\wps-toolSupport.war
- portal\lib\commerce\ejb\commerce.jar
- portal\lib\netuix\ejb\netuix.jar
- portal\lib\netuix\ejb\pipeline.jar
- portal\lib\netuix\ejb\prefs.jar
- portal\lib\portal\ejb\portal.jar
- portal\lib\wps\ejb\wps.jar
- p13n\lib\p13n_ejb.jar
- p13n\lib\datasync.war

6. Make the following changes in the <*ENT-APP*>/<*WEB-APP*>/WEB-INF/lib directory for each of the web applications in your enterprise application:

- delete the p13n_servlet.jar if it exists
- replace ad_taglib.jar with weblogic81/portal/lib/wps/web/ad_taglib.jar
- replace cm_taglib.jar with weblogic81/portal/lib/wps/web/cm_taglib.jar
- replace ph_taglib.jar with weblogic81/portal/lib/wps/web/ph_taglib.jar
- replace pz_taglib.jar with weblogic81/portal/lib/wps/web/pz_compat_taglib.jar and weblogic81/portal/lib/wps/web/pz_taglib.jar
- replace p13n_servlet.jar with weblogic81/portal/lib/wps/web/wps_servlet.jar

- replace `webflow_servlet.jar` with `weblogic81/portal/lib/netuix/web/webflow_servlet.jar`
- replace `webflow_taglib.jar` with `weblogic81/portal/lib/netuix/web/webflow_taglib.jar`
- add or replace `cat_taglib.jar` with `weblogic81/portal/lib/commerce/web/cat_taglib.jar`
- add `weblogic81/portal/lib/commerce/web/eb_taglib.jar`
- add/replace `productTracking_taglib.jar` with `weblogic81/portal/lib/commerce/web/productTracking_taglib.jar`
- replace `dam_taglib.jar` with `weblogic81/portal/lib/portal/web/dam_taglib.jar`
- replace `ent_taglib.jar` with `weblogic81/portal/lib/portal/web/ent_taglib.jar`
- replace `portal_servlet.jar` with `weblogic81/portal/lib/portal/web/portal_servlet.jar`
- replace `portal_taglib.jar` with `weblogic81/portal/lib/portal/web/portal_taglib.jar`
- replace `portlet_taglib.jar` with `weblogic81/portal/lib/portal/web/portlet_taglib.jar`
- replace `ren_taglib.jar` with `weblogic81/portal/lib/portal/web/ren_taglib.jar`
- replace `res_taglib.jar` with `weblogic81/portal/lib/portal/web/res_taglib.jar`
- replace `util_taglib.jar` with `weblogic81/portal/lib/portal/web/util_taglib.jar`
- replace `visitor_taglib.jar` with `weblogic81/portal/lib/portal/web/visitor_taglib.jar`
- replace `vum_taglib.jar` with `weblogic81/portal/lib/portal/web/vum_taglib.jar`
- replace `es_taglib.jar` with `weblogic81/p13n/lib/es_taglib.jar`

- replace i18n_taglib.jar with weblogic81/p13n/lib/i18n_taglib.jar
- replace ps_taglib.jar with weblogic81/p13n/lib/ps_taglib.jar
- replace tracking_taglib.jar with weblogic81/p13n/lib/tracking_taglib.jar
- replace um_taglib.jar with weblogic81/p13n/lib/um_taglib.jar
- replace weblogic-tags.jar with weblogic81/server/ext/weblogic-tags.jar
- If you are upgrading sampleportal, replace xmlx-tags.jar (used by the moreNews portlet) with weblogic81/samples/server/examples/build/examplesWebApp/WEB-INF/lib/xmlx-tags.jar

7. Make the following changes in the application.xml file in the <ENT-APP>/META-INF/ directory:

- Replace <web-uri>tools</web-uri> with <web-uri>tools700.war</web-uri>
- Replace <web-uri>datasync</web-uri> with <web-uri>datasync.war</web-uri>
- Replace <web-uri>toolSupport</web-uri> with <web-uri>toolSupport.war</web-uri>

Note: Replace the <web-uri>, not the <context-root>.

- Remove references to the following BEA-supplied EJB modules (if present):
  - document.jar
  - ejbadvisor.jar
  - events.jar
  - mail.jar
  - pipeline.jar
  - placeholder.jar
  - property.jar
  - rules.jar
  - usermgmt.jar
  - catalogws.jar
  - customer.jar
  - ebusiness.jar
  - campaign.jar

- Retain the following BEA-supplied EJB module listing (if present):
  - portal.jar
- Add the following EJB module references:
  - netuix.jar
  - pipeline.jar
  - prefs.jar
  - p13n_ejb.jar
  - wps.jar
  - content_repo.jar
  - content.jar
  - commerce.jar

8. Make the following changes in the application-config.xml file in the <ENT-APP>/META-INF/ directory:

- If using commerce, replace any existing AttributeLoader reference with this one:

```
<AttributeLoader Name="AttributeLoader"
RequestLoaders="com.bea.campaign.ShoppingCartAttributeLoader" />
```

- Within the ApplicationConfiguration node, after the last DocumentConnectionPool entry, add the following entry:

Listing 2-7 DocumentConnectionPool Entry

```
<ContentManagement Name="ContentManagement">
<ContentStore Name="adapter"
ClassName="com.bea.p13n.content.adapter.RepositoryImpl"
Properties="CONTENT_MANAGER_HOME=${APPNAME}.BEA_personalization.DocumentMa
nager" />
</ContentManagement>
<CommercePipelineComponentSupport
JdbcPoolName="weblogic.jdbc.jts.commercePool" />
```

Content Management Configuration

The SamplePortal application provided with WebLogic Portal 7.0 uses a custom Document Provider Interface (DPI) integration for content management. If the enterprise application you are upgrading also uses such an integration, this generally implies that you have one or more JSPs using code similar to that shown in Listing 2-8. If this is the case, perform the steps in "Step 1: Change the Content Management application-config.xml File" on page 2-19.

Listing 2-8 Custom Content Management JSP Code

```
<pz:contentSelector ... contentHome="java:comp/env/ejb/NewsletterManager"
<-- NOTE this is not the standard value of
"<%=ContentHelper.DEF_DOCUMENT_MANAGER_HOME%>" />
```

Step 1: Change the Content Management application-config.xml File

Make the following modifications to support this Content Management feature:

1. For each additional document provider, add a `<ContentStore>` section to the `<ContentManagement>` node. The `<jndi-name>` for the document provider can be found in `weblogic-ejb-jar.xml`. For sampleportal, the entry is as follows:

```
<jndi-name>${APPNAME}.BEA_portal_examples.NewsletterDocumentManager</jndi-name>
```

2. Set the `CONTENT_MANAGER_HOME` property of the `<ContentStore>` to be the JNDI name used to look up the `documentprovider`. For the sampleportal, the entry is as follows:

```
Properties="CONTENT_MANAGER_HOME=${APPNAME}.BEA_portal_examples.NewsletterDocumentManager"
```

For sampleportal, add this as a second `<ContentManagement>` element, as shown in Listing 2-9:

Listing 2-9 The second <ContentManagement> element for sampleportal

```
<ContentStore
Name="newsletters"
ClassName="com.bea.p13n.content.adapter.RepositoryImpl"
Properties="CONTENT_MANAGER_HOME=${APPNAME}.BEA_portal_examples.Newsletter
DocumentManager" />
```

Step 2: Edit the web.xml File

For each web application, make the following change to the `web.xml` file in the `<ENT-APP>/<WEB-APP>/WEB-INF/` directory:

Replace the code in Listing 2-10 with that in Listing 2-11.

Listing 2-10 pz_taglib.jar reference

```
<taglib>
<taglib-uri>pz.tld</taglib-uri>
<taglib-location>/WEB-INF/lib/pz_taglib.jar</taglib-location>
</taglib>
```

Listing 2-11 pz_compat_taglib.jar reference

```
<taglib>
<taglib-uri>pz.tld</taglib-uri>
<taglib-location>/WEB-INF/lib/pz_compat_taglib.jar</taglib-location>
</taglib>
```

Step 3: Edit the config.xml File

This section lists two modifications to the `config.xml` file; one for commerce support, the other for every enterprise application.

If the enterprise application uses commerce features, edit the `config.xml` file, inserting the following StartupClass section above any application listings, and setting the Target to the servername, as shown in Listing 2-12.

Listing 2-12 StartupClass Entry for Commerce Support

```
<StartupClass
ClassName="com.beasys.commerce.ebusiness.security.KeyBootstrap"
LoadBeforeAppActivation="true"
FailureIsFatal="false"
```

```
Name="Commerce KeyBootstrap"
Targets="portalServer"/>
```

Add a reference to the enterprise application(s), complete with the correct path. Examples using sample applications that shipped with WebLogic Portal 7.0 are shown in Listing 2-13, Listing 2-14, Listing 2-15, and Listing 2-16.

The following notes apply to Listing 2-13, Listing 2-14, Listing 2-15, and Listing 2-16:

- References to the enterprise applications depend on the application itself. Consult the `config.xml` file in your existing WebLogic Portal 7.0 application, as well as the `META-INF/application.xml` file you modified above to obtain the correct values for your particular application.
- If the application contains an EJB which is built during the application build, comment out the EJB deployment for now. For example, see the commented EJBComponent in Listing 2-13.
- The `portal.jar` is present only for certain enterprise applications.
- If the existing WebLogic Portal 7.0 application used commerce, then update the `paymentWSApp` and `taxWSApp` applications, setting the deployed="true".
- Be sure to verify paths.

Listing 2-13 SAMPLEPORTAL config.xml File

```
<Application Deployed="true" Name="sampleportal"
Path="D:\bea81\weblogic81\samples\portal\sampleportal" TwoPhase="true">
<ApplicationConfiguration Name="sampleportal"
Targets="portalServer" URI="META-INF/application-config.xml"/>
<ConnectorComponent Name="BlackBoxNoTx" Targets="portalServer"
URI="BlackBoxNoTx.rar"/>
<EJBComponent Name="commerce" Targets="portalServer" URI="commerce.jar"/>
<EJBComponent Name="p13n_ejb" Targets="portalServer" URI="p13n_ejb.jar"/>
<EJBComponent Name="portal" Targets="portalServer" URI="portal.jar"/>
<!-- <EJBComponent Name="sampleportal" Targets="portalServer"
URI="sampleportal.jar"/>  -->
<EJBComponent Name="wps" Targets="portalServer" URI="wps.jar"/>
<EJBComponent Name="pipeline" Targets="portalServer" URI="pipeline.jar"/>
<EJBComponent Name="portalmanager" Targets="portalServer" URI="netuix.jar"/>
<EJBComponent Name="prefs" Targets="portalServer" URI="prefs.jar"/>
<EJBComponent Name="content" Targets="portalServer" URI="content.jar"/>
```

```
<EJBComponent Name="contentrepo" Targets="portalServer"
URI="content_repo.jar"/>
<WebAppComponent Name="datasync" ServletReloadCheckSecs="300"

Targets="portalServer" URI="datasync.war"/>
<WebAppComponent Name="defaultWebApp"
ServletReloadCheckSecs="300" Targets="portalServer" URI="defaultWebApp"/>
<WebAppComponent Name="sampleportal"
ServletReloadCheckSecs="300" Targets="portalServer" URI="sampleportal"/>
<WebAppComponent Name="toolSupport" ServletReloadCheckSecs="300"
Targets="portalServer" URI="toolSupport.war"/>
<WebAppComponent Name="tools" ServletReloadCheckSecs="300"
Targets="portalServer" URI="tools700.war"/>
</Application>
<Application Deployed="true" Name="wlpDocsApp"
Path="D:\bea81\weblogic81\portal\lib"
StagedTargets="portalServer" TwoPhase="true">
<WebAppComponent IndexDirectoryEnabled="false" Name="wlpDocs"
ServletReloadCheckSecs="300" Targets="portalServer" URI="wlpDocs.war"/>
</Application>
```

Listing 2-14 P13NApp config.xml File

```
<Application Deployed="true" Name="p13nApp"
Path="D:\weblogic81\samples\portal\p13nApp" TwoPhase="true">
<ApplicationConfiguration Name="p13nApp" Targets="portalServer"
URI="META-INF/application-config.xml"/>
<EJBComponent Name="commerce" Targets="portalServer" URI="commerce.jar"/>
<EJBComponent Name="p13n_ejb" Targets="portalServer" URI="p13n_ejb.jar"/>
<EJBComponent Name="wps" Targets="portalServer" URI="wps.jar"/>
<EJBComponent Name="pipeline" Targets="portalServer" URI="pipeline.jar"/>
<EJBComponent Name="portalmanager" Targets="portalServer" URI="netuix.jar"/>
<EJBComponent Name="prefs" Targets="portalServer" URI="prefs.jar"/>
<EJBComponent Name="content" Targets="portalServer" URI="content.jar"/>
<EJBComponent Name="contentrepo" Targets="portalServer"
URI="content_repo.jar"/>
<WebAppComponent Name="datasync" ServletReloadCheckSecs="300"
Targets="portalServer" URI="datasync.war"/>
<WebAppComponent Name="defaultWebApp"
ServletReloadCheckSecs="300" Targets="portalServer" URI="defaultWebApp"/>
<WebAppComponent Name="p13n" ServletReloadCheckSecs="300"
Targets="portalServer" URI="p13n"/>
<WebAppComponent Name="toolSupport" ServletReloadCheckSecs="300"
Targets="portalServer" URI="toolSupport.war"/>
<WebAppComponent Name="tools" ServletReloadCheckSecs="300"
Targets="portalServer" URI="tools700.war"/>
```

```
</Application>
<Application Deployed="true" Name="wlpDocsApp"
Path="D:\weblogic81\portal\lib"
StagedTargets="portalServer" TwoPhase="true">
<WebAppComponent IndexDirectoryEnabled="false" Name="wlpDocs"
ServletReloadCheckSecs="300" Targets="portalServer" URI="wlpDocs.war"/>
</Application>
```

Listing 2-15 WLCSApp config.xml File

```
<Application Deployed="true" Name="wlcsApp"
Path="D:\weblogic81\samples\portal\wlcsApp" TwoPhase="true">
<ApplicationConfiguration Name="wlcsApp" Targets="portalServer"
URI="META-INF/application-config.xml"/>
<EJBComponent Name="commerce" Targets="portalServer" URI="commerce.jar"/>
<EJBComponent Name="p13n_ejb" Targets="portalServer" URI="p13n_ejb.jar"/>
<!-- <EJBComponent Name="wlcsSample" Targets="portalServer"
URI="wlcsSample.jar"/> -->
<EJBComponent Name="wps" Targets="portalServer" URI="wps.jar"/>
<EJBComponent Name="pipeline" Targets="portalServer" URI="pipeline.jar"/>
<EJBComponent Name="portalmanager" Targets="portalServer" URI="netuix.jar"/>
<EJBComponent Name="prefs" Targets="portalServer" URI="prefs.jar"/>
<EJBComponent Name="content" Targets="portalServer" URI="content.jar"/>
<EJBComponent Name="contentrepo" Targets="portalServer"
URI="content_repo.jar"/>
<WebAppComponent Name="datasync" ServletReloadCheckSecs="300"
Targets="portalServer" URI="datasync.war"/>
<WebAppComponent Name="defaultWebApp"
ServletReloadCheckSecs="300" Targets="portalServer" URI="defaultWebApp"/>
<WebAppComponent Name="wlcs" ServletReloadCheckSecs="300"
Targets="portalServer" URI="wlcs"/>
<WebAppComponent Name="toolSupport" ServletReloadCheckSecs="300"
Targets="portalServer" URI="toolSupport.war"/>
<WebAppComponent Name="tools" ServletReloadCheckSecs="300"
Targets="portalServer" URI="tools700.war"/> </Application>
<Application Deployed="true" Name="wlpDocsApp"
Path="D:\weblogic81\portal\lib"
StagedTargets="portalServer" TwoPhase="true">
<WebAppComponent IndexDirectoryEnabled="false" Name="wlpDocs"
ServletReloadCheckSecs="300" Targets="portalServer" URI="wlpDocs.war"/>
</Application>
```

Listing 2-16 StockPortal config.xml File

```
<Application Deployed="true" Name="stockportal"
Path="D:\weblogic81\samples\portal\portal" TwoPhase="true">
<ApplicationConfiguration Name="portal" Targets="portalServer"
URI="META-INF/application-config.xml"/>
<EJBComponent Name="commerce" Targets="portalServer" URI="commerce.jar"/>
<EJBComponent Name="p13n_ejb" Targets="portalServer" URI="p13n_ejb.jar"/>
<!--        <EJBComponent Name="stockportal" Targets="portalServer"
URI="stockportal.jar"/> -->
<EJBComponent Name="portal" Targets="portalServer" URI="portal.jar"/>
<EJBComponent Name="wps" Targets="portalServer" URI="wps.jar"/>
<EJBComponent Name="pipeline" Targets="portalServer" URI="pipeline.jar"/>
<EJBComponent Name="portalmanager" Targets="portalServer" URI="netuix.jar"/>
<EJBComponent Name="prefs" Targets="portalServer" URI="prefs.jar"/>
<EJBComponent Name="content" Targets="portalServer" URI="content.jar"/>
<EJBComponent Name="contentrepo" Targets="portalServer"
URI="content_repo.jar"/>
<WebAppComponent Name="datasync" ServletReloadCheckSecs="300"
Targets="portalServer" URI="datasync.war"/>
<WebAppComponent Name="defaultWebApp"
ServletReloadCheckSecs="300" Targets="portalServer" URI="defaultWebApp"/>
<WebAppComponent Name="stockportal" ServletReloadCheckSecs="300"
Targets="portalServer" URI="stockportal"/>
<WebAppComponent Name="toolSupport" ServletReloadCheckSecs="300"
Targets="portalServer" URI="toolSupport.war"/>
<WebAppComponent Name="tools" ServletReloadCheckSecs="300"
Targets="portalServer" URI="tools700.war"/>
<WebAppComponent Name="workshop" ServletReloadCheckSecs="0"
Targets="portalServer" URI="workshop"/>
</Application>
```

Modify Code

Search the code from the existing WebLogic Portal 7.0 application for calls to the following class:

`com.bea.p13n.util.jdbc.SequencerFactory createSequencer( String, DataSource)` method. Any calls to this method must be updated as follows:

Replace the code in Listing 2-17 with that in Listing 2-18.

Listing 2-17 A Call to createSequencer Class: Before

```
sequencer = SequencerFactory.createSequencer("ORDER_LINE_ID_SEQUENCE",
getDataSource());
```

Listing 2-18 A Call to createSequencer Class: After

```
sequencer = SequencerFactory.createSequencer( "ORDER_LINE_ID_SEQUENCE" );
```

Note: Exceptions are different, so you might also need to update any enclosing try/catch blocks.

Step 1: Upgrade Web Service Client Proxies

Find all web service proxy (client) code used by the enterprise application or by the Web application. Usually there is a `.wsdl` file, a `*Codec.java` or `*_Stub.java`**.**

- Delete the Web service proxy code such as:
  - `*Codec.java`
  - `*_Stub.java`
  - `*_Impl.java`
  - `*ServicePort.java`
  - `*Request.java`
  - `*RequestElement.java`
  - `*Response.java`
  - `*ResponseElement.java`
  - `*Service.java`

This section uses concrete examples of steps taken to make the `WLCSApp` compliant with the new framework.

- Delete the following files in the **src/examples/wlcs/sampleapp/payment** directory:
  - `ArrayOfResponseElementCodec.java`
  - `PaymentService.java`
  - `PaymentService.xml`

  - PaymentService_Impl.java
  - PaymentServicePort.java
  - PaymentServicePort_Stub.java
  - PSRequest.java
  - PSRequestCodec.java
  - PSResponse.java
  - PSResponseCodec.java
  - PSResponseElement.java
  - PSResponseElementCodec.java
- Delete the following file in the src/language_builtins/lang directory:
  - ArrayOfStringCodec.java
- Delete the following files in the **src/examples/wlcs/sampleapp/tax** directory:
  - ArrayOfAVSResponseElementCodec.java
  - ArrayOfTaxRequestElementCodec.java
  - ArrayOfTaxResponseElementCodec.java
  - AVSRequest.java
  - AVSRequestCodec.java
  - AVSResponse.java
  - AVSResponseCodec.java
  - AVSResponseElement.java
  - AVSResponseElementCodec.java
  - TaxRequest.java
  - TaxRequestCodec.java
  - TaxRequestElement.java
  - TaxRequestElementCodec.java
  - TaxResponse.java
  - TaxResponseCodec.java
  - TaxResponseElement.java
  - TaxResponseElementCodec.java
  - TaxService.java
  - TaxService.xml

  - TaxService_Impl.java
  - TaxServicePort.java
  - TaxServicePort_Stub.java

- Make a backup of the source tree.
- Start WebLogic server.

Step 2: Change Web Service Proxies

If you modified the web service proxies in the WebLogic Portal 7.0 domain, you need to do the same modifications in the new domain. In any case, the files will need to be re-generated because the generated proxy jar file code has changed.

Note: The proxy code needs to be in a separate jar in the `APP-INF/lib` directory to work at runtime. It will not work if you merely compile it and place it in `wlcsSample.jar`. Therefore, you must generate a proxy jar file, and then manually replace any changed proxy files.

Here is an example of a target that generates proxy jar files—add an entry such as that shown in Listing 2-19 to your build file.

Listing 2-19 Web Service Proxy Rewrite Entry

```
<target name="stubgen" >
<property name="taxWsdlUrl"
value="http://localhost:7501/taxws/TaxService?WSDL" />
<property name="payWsdlUrl"
value="http://localhost:7501/payws/PaymentService?WSDL" />
<clientgen wsdl="${taxWsdlUrl}" packageName="examples.wlcs.sampleapp.tax"
clientJar="taxwsproxy.jar" />
<clientgen wsdl="${payWsdlUrl}"
packageName="examples.wlcs.sampleapp.payment"
clientJar="paywsproxy.jar" />
</target>
```

Upgrade Application-Sync Files

Follow the "upgrade application-sync" steps in "Step 3: Upgrade Application-Sync Files" on page 3-9.

Final Adjustments to the Domain

This set of steps requires at least one enterprise application to be upgraded and deployed, because it requires the WebLogic 7.0 Portal Administration Tools.

Add Users to PortalSystemAdministrators Group

Add all members of the WebLogic 7.0 Portal SystemAdministrator group to the PortalSystemAdministrators group.

This procedure is explained using sampleportal.

1. Launch the server in your 8.1 compatibility domain.
2. Open Weblogic Portal Administration Tools at http://localhost:7501/sampleportalTools and login as weblogic/weblogic, or whatever system user ID and password you used to create the 8.1 compatibility domain.
3. In User Management, under Groups, select **PortalSystemAdministrators**.
4. Select the users in the WebLogic Portal 7.0 **SystemAdministrator** group, and add them to **PortalSystemAdministrators**.
5. Log into the WebLogic Server Console, choose **Security→Realms→myRealms→Global Roles**, and click **Configure a New Global Role**.
   - administrator
   - demosa1
   - demosa2
6. Configure the new global role as described in Creating Global Roles, at:

   `http://e-docs.bea.com/wls/docs81/secwlres/secroles.html#1219839`
7. Save the changes and shut down the server.

Upgrade CustomerRole in SSPI to Point to RDBMS Groups

If the existing WebLogic 7.0 domain contained an application that uses commerce, the CustomerRole security role must be associated with the `wlcs_customer` RDBMS group. This modification is necessary because normally the CustomerRole security role is associated with the `wlcs_customer` LDIFT group, and therefore won't perform correctly with members of the `wlcs_customer` RDBMS group.

Note: To perform this configuration, the assignment must be removed from the default LDIFT files. This means that if you delete the server directory (which includes LDAP information), you will need to do these steps again.

To associate the CustomerRole security role with the `wlcs_customer` RDBMS group, take the following steps:

1. Edit the compatibility domain's `DefaultRoleMapperInit.ldift` file, deleting the following section:

   ```
   dn: cn=::CustomerRole,ou=ERole,ou=@realm@,dc=@domain@

   objectclass: top

   objectclass: ERole

   cn: ::CustomerRole

   EExpr:: Z3dsY3NfY3VzdG9tZXIK
   ```

2. Edit the compatibility domain's **security.xml** file, deleting the following sections as shown in Listing 2-20 and Listing 2-21 below.

Listing 2-20 ns1:group Entry

```
<ns1:group name="wlcs_customer">
<ns1:roleMemberOf ref="CustomerRole"/>
</ns1:group>
```

Listing 2-21 ns1:role Entry

```
<ns1:role name="CustomerRole" description="View/modify the wlcs customer settings"/>
```

3. Delete the **portalServer** directory inside the compatibility domain. This directory is named **<yourEnterpriseApplicationName>Server**.
4. Start the WebLogic Portal Server.
5. Log into the WebLogic Server console, and click **Configure a new global role**.
6. Name the new role **CompatibilityCustomerRole**, and click **Apply**.
7. From the Conditions tab, select **Caller is a member of the group** and click **Add**.
8. Enter group name **wlcs_customer** and click **Add**, **OK**, then **Apply**.
9. Shut down the server.

Users and Groups Specified in fileRealm.properties

You must add any system-level users and groups, which under 7x were specified in fileRealm.properties, to the authentication provider used by WebLogic Portal 8.1.

Note: This step is not needed for sampleportal, wlcsApp, p13nApp, or stockportal.

Unlike WebLogic Portal 7.0, which allowed users and groups to be stored in both fileRealm (where system users and groups were stored) and the main store (where typical users and groups were stored), WebLogic Portal 8.1 works with users and groups in the single authentication provider.

Therefore, if you want to support system users and groups (weblogic, Operators, Monitors, and so on) in the portal, you will need to move these users over to the authentication provider. This applies to any system users or groups in the fileRealm to which you want to provide portal access and portal tool access in the new Portal Compatibility Domain.

CHAPTER 3

Upgrading to WebLogic Portal 8.1

Upgrading from Compatibility Mode to WebLogic Portal 8.1

The first part of this section gives instructions on moving an application that has been upgraded to run inside the WebLogic Portal Compatibility Domain so that it will run in a regular WebLogic Portal 8.1 domain.

The remainder of this section explains WebLogic Portal 8.1 features and architectural details pertinent to upgrade issues. These aspects of the new Portal framework include Webflow support, Portal services, Content Management, and Struts support.

Before You Begin

It is not possible to move directly from a WebLogic Portal 7.0 SP2 Domain to an out-of-the-box WebLogic Portal 8.1 domain. This procedure requires a Portal Web application running in a Compatibility Domain. For detailed instructions on setting up a compatibility domain, see the Creating a Portal Compatibility Domain.

1. Run the 'upgradePortalFiles.cmd' utility and place your upgraded portals and portlets in your application.
2. While no particular location is required, the upgrade tool will arrange portals and portlets such that the portlet references in the portal are in the correct location so the easiest thing to do is copy the output from the tool directly to the `web-app` directory.
3. Ensure that the JSPs referenced by the portlets are in the correct relative location.

Procedure for Upgrading from Compatibility

To convert an application running in a Portal Compatibility Domain to one that can run in an out-of-the-box WebLogic Portal 8.1 domain, take the following steps:

1. Create new domain using the Domain Wizard, selecting the Express option.
2. Create a new application directory in your new domain—in `beaApps`, for example.
3. Create a new Portal Application.
4. Right-click **Application** in the Workshop Application tab. Choose **Import** and use the wizard to import the Web application you re-hosted using the steps in "Hosting 7.0 Applications in 8.1 Compatibility Domain." In the dialog, choose **Portal Web Project,** then browse to the web application. Make sure the "Copy into Application directory" option is checked. You are prompted with the message, "Files or directories required by this project type are not present. Would you like to have Workshop update your project?" Select **yes**.
5. Use WebLogic Workshop to delete the following files:
   - `portal_taglib.jar`
   - `portlet_taglib.jar`
   - `portal_servlet.jar`

   These JARs contain com.bea.portal classes that are not supported in WebLogic Portal 8.1. Review and rewrite your applications as needed to discontinue use of these classes.
6. If your application uses pipeline components, add support for this functionality. Right-click **Application** in the Application tab. Choose **Install→Pipeline Services**.
7. If the original application had Webflow and you followed the Compatibility Domain upgrade instructions, this new project should already include Webflow support. If it does not, and you need Webflow support, add it by right-clicking the project (webApp) and selecting **Install→Webflow Taglibs**.
8. If the application requires any EJB **.jar** or **.war** files you created, add them by right-clicking **Modules** in the Application pane, selecting **Add Module,** browsing for the files, and clicking **Open**.

Note: You should not need `tools700.war` or `toolSupport.war`. This procedure enables the use of WebLogic 8.1 Administration Portal against the application.

9. Import the data files from the `META-INF/data` sub-directories of the Portal Compatibility Domain application to the `META-INF/data` directory in the new domain. The data directory must be imported component-by-component.
   - To import a new data sub-directory, right-click the data directory in the Workshop Application pane and select **Import.** Then browse to the directory from the Portal Compatibility Domain and click **Import**.
   - To merge individual files into an existing data sub-directory, (campaigns, for example) right-click the Workshop Application pane and select **Import.** Then browse to the directory from the Portal Compatibility Domain and click **Import**. The old portlet files are not needed by the application, but they can be imported to use for reference.
10. Copy the upgraded portals and portlets into the web application. Place the portal directly in the `<WEB-APP>` directory and the portlets in the `<WEB-APP>/portlets` directory, which you might need to create.
    - This placement is not required for the application to function; however, the upgradePortalFiles tool assumes this relative location. Therefore, if you choose a different organization, you might need to modify the contentUri attribute of the netuix:portletInstance tags in the portal. Check any JSP references in the upgraded portlet files and adjust them if the specified path does not match the actual location of the JSP. WebLogic Workshop will synchronize with the file system, and the upgraded portal(s) and portlets should be visible in the web app in the Application pane.
11. Open the upgrade portal(s) in WebLogic Workshop, and correct the layout as desired; the upgrade tool chooses a simple single column layout that might not be suitable for your portal.
12. Start the server. The server will generate error messages including “EJB not deploying.” This is most likely because the new connections pools and data sources do not match those from the original application. We'll fix this next.
13. Using the `config.xml` file from the Compatibility Domain as a guide, create new connection pool(s) and data source(s) as needed for your application. With the server running, go to the WebLogic Console and create pool(s) and source(s) to match those in the `config.xml` of the earlier version of your application.
14. Deploy the application using the WebLogic Console.
15. Make any additions to the server classpath that might be required by your application.
16. If errors are being generated, you might need to search for references to com.bea.portal classes. These need to be replaced with references to new classes.

17. (Optional) The next step is to replace Webflows with Page Flows and the WebflowServlets that are produced by the portal upgrade tool with standard WebLogic 8.1 portals. Because Webflow support can easily be added to applications in WebLogic 8.1, this step is optional.

Adding Catalog Administration

You can add WebLogic Portal catalog administration functionality to your portal applications. The following procedure allows you to create and manage catalog categories and content items. You can also create and manage catalog property sets in the WebLogic Workshop Portal Extensions and manage them in the online catalog administration tools that you add with this procedure.

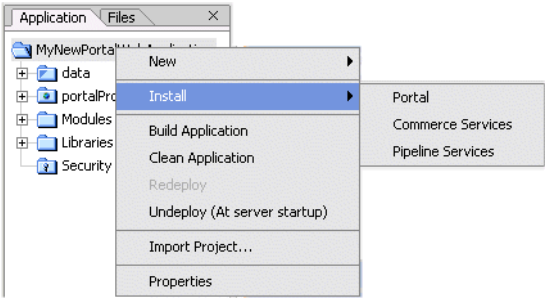
1. Copy the tools700.war into the J2EE Application root folder.

   From the **<**`WebLogic Platform 8.1`**>**`\portal\lib` directory, copy the `tools700.war` file into the enterprise directory for your application. For example, to add this functionality to the sample portal application, place this file in `samples\portal\portalApp`. The next time this application is opened in WebLogic Workshop, the catalog administration tools are deployed automatically.

   Note: The location, as well as the Web Application name, are changed in WebLogic Portal Version 8.1.

2. If commerce services haven't been installed in the application, right-click the Application directory and select **Install→Commerce Services**.

Figure 3-3 Adding Commerce Service



3. Copy the Webflow files supporting the commerce administration tools into the new application:

a. In the `\samples\portal\portalApp\META-INF\data` directory in your WebLogic 8.1 installation directory, create a directory called `webapps`, and within that directory, create one called `tools700`.

b. From the `\beaApps\sampleportal-project\application-sync\webapps\tools\` directory in the WebLogic Portal 7.0 SP2 domain, copy the tools Webflow files into the newly-created `/webapps/tool700` directory.

c. Rename this directory to the context root of your `tools700.war` file. the default is `tools700`.

4. To use these tools, open a Web browser and navigate to http://<hostname>:<port>/tools700 and click **Catalog Management**.

Group Portals

Group portal definitions are not upgraded: in WebLogic Portal 8.1, the corresponding mechanism is the Desktop, by which a bundle of portal components is presented to a specified audience.

Struts Support

This section provides general information for developing Struts applications in WebLogic Portal 7.0, in preparation for Struts-based Page Flows in the 8.1 release. General guidelines are also provided for upgrading a Struts-based application to run within WebLogic Portal 8.1.

Struts Applications in WebLogic Portal 7.0

WebLogic Portal 8.1 supports Struts 1.1. Keep the following guidelines in mind as you develop Struts-based applications:

- Restrict Struts applications to the portlet level. Struts should not be used as a page framework or in the portal framework.
- Use Struts modules (sub-apps), keep all pages for modules in a single directory, and do not share pages across modules.
- Use request scoped action forms, and avoid the use of Struts plugins such as Tiles.

Support for Struts in WebLogic Portal 8.1

You can edit Struts-based application configuration files in WebLogic Workshop "source view." If you are familiar with Struts, you can edit and develop against the struts-config XML file and have these updates merged to a Java Page Flow. These updates are merged to the struts-config

XML file generated from the Java Page Flow (`.jpf`) file. You can also merge existing Struts resources into a Page Flow file.

Note: Running Struts applications in WebLogic Portal requires changes to JSP tags to handle URL rewriting.

Upgrading Struts Applications to Page Flows in WebLogic Portal 8.1

Page Flow applications are built on Struts 1.1. WebLogic Workshop provides visual design tools and a two-way editing environment for developing Page Flows, which in turn provide a single-file model for development.

Note: Upgrading JSP pages from a Struts application to Page Flow will require some changes to JSP tags in order to take advantage of Page Flow features.

JSP Tag Replacements

Appendix E, "Framework Reference for Portal Upgrades from 7.0 to 8.1," describes JSP tags deprecated in WebLogic Portal 8.1, suggesting replacement tags or other refactoring strategies where possible.

Upgrading Database and Metadata Files from 7.0 to 8.1

This section provides general information concerning upgrade procedures for WebLogic Portal database and metadata files. The following topics are included in this section:

- Step 1: Upgrade PointBase Triggers
- Step 2: Upgrade WebLogic Portal 7.0 Schema
- Step 3: Upgrade Application-Sync Files
- Step 4: Upgrade Portal and Portlet Files
- Step 5: Upgrade ENTITLEMENT_RULESET Records
- Step 6: Upgrading Existing Behavior Tracking Data

Table 3 -1 describes the situations where you would use each step.

Table 3 -1 Upgrade Scenarios

Upgrade Strategy	Perform Steps
Using PointBase, moving to a WebLogic Portal 8.1 Domain	1 through 4
Not using PointBase, moving to a WebLogic Portal 8.1 Domain	2 through 4
Using PointBase, moving WebLogic Portal 7.0 application to a Portal Compatibility Domain	1, 2, 3 and 5
Not using PointBase, moving WebLogic Portal 7.0 application to a Portal Compatibility Domain	2, 3 and 5

To use your existing PointBase database with the PointBase version shipped with WebLogic 8.1, perform Step 1 below. In order to upgrade portions of Portal 7.0 to Portal 8.1, perform Steps 2 through 4. And, finally, if you are planning to run in Compatibility Mode, perform Step 5.

Note: None of these steps require that WebLogic Server be up and running, and, as a result, no XA JDBC driver should be used in connecting to the database.

Step 1: Upgrade PointBase Triggers

WebLogic 8.1 includes a more current release of PointBase (4.4). PointBase rewrote their implementation of database triggers in PointBase 4.3. As a result, you must perform the following steps to upgrade your PointBase database for use in a Version 8.1 WebLogic domain.

1. Shut down the WebLogic Server.
2. Make backup copies of the 7.0 PointBase database files. This can be accomplished by simply copying the two data files (`*$1.wal` and `*.dbn`) found in the `..\<7.0 domain name>\pointbase` directory. Be sure to make backup copies or the data files for all pertinent domains.
3. Change directories to the 7.0 domain directory of interest and start the PointBase console by running the `startPBConsole db_settings.properties` file in the following directory:

   `..\bea70\weblogic70\portal\bin\win32\`
4. Open the following file in the PointBase console and execute the SQL by choosing **SQL** → **Execute All**:

   `..\bea81\weblogic81\portal\upgrade\drop_70_triggers.sql`

5. Shut down the PointBase console.
6. Copy the 7.0 PointBase data files from the `..\<7.0 domain name>\pointbase` directory to the 8.1 upgrade directory:

   ```
   ..\bea81\weblogic81\portal\upgrade\pointbase
   ```

7. Change directories to: `..\bea81\weblogic81\portal\upgrade` and start the PointBase server and PointBase console:

   ```
   ..\bea81\weblogic81\common\bin\startPointBase.cmd
   -ini=pointbase\pointbase.ini
   ```

   ```
   ..\bea81\weblogic81\common\bin\startPointBaseConsole
   ```

8. Open the following file in the PointBase console and execute the SQL by going to **SQL** → **Execute All**:

   ```
   ..\bea81\weblogic81\portal\upgrade\create_70_triggers.sql
   ```

9. Shut down the PointBase console and PointBase server.

Step 2: Upgrade WebLogic Portal 7.0 Schema

Complete the following steps to make your 7.0 database schema 8.1 compliant:

1. Back up your 7.0 database.
2. Change directories to `..\weblogic81\portal\upgrade`.
3. Update `db_settings.properties` with your 7.0 database settings.
4. Run the following script: `upgrade_db_schema_to_81`
5. Review the log file, `upgrade_db_schema_to_81.log`, to verify that the upgrade was successful.
6. For Sybase databases, run the upgrade script for 7.0 to 8.1.

   The WebLogic Portal 7.0 upgrade script is located at `bea\weblogic700\portal\db\sybase\125\migrate\migrate_to_125.sql`

7. For PointBase; copy the `..\weblogic81\portal\upgrade\pointbase\*$1.wal` and `*.dbn` files to your 8.1 domain. If your 8.1 domain is not named `wlpCompatability`, you must rename the `*$1.wal` and `*.dbn` files to match the PointBase database name used in your domain.

The database is now upgraded to WebLogic Portal 8.1.

Note: For PointBase you will see 2 errors (example below). You can ignore these errors, as they are generated while attempting to re-create the user WEBLOGIC:

```
SQL> Error Message: User WEBLOGIC already exists in the database.
```

Step 3: Upgrade Application-Sync Files

A number of files require the xsi:schemaLocation to be updated as a result of an upgrade to Xerces. These files typically reside in a directory similar to:

```
..\bea70\weblogic700\user-projects\portalDomain\beaApps\portalApp-project\
application-sync
```

1. Copy the 7.0 application-sync directory and its subdirectories and files to the 8.1 domain location. The application-sync should be a peer to `..\META-INF\data`.

   For example:

   ```
   ..\bea81\user_projects\applications\portalApp\META-INF\application-sync
   ```

   ```
   ..\bea81\user_projects\applications\portalApp\META-INF\data
   ```

   From the `..\bea81\weblogic81\portal\upgrade` directory, run the script by typing the following at the command line:

   ```
   UpgradeApplicationSyncFiles c:\bea81\user_projects\applications\
   testApp\META-INF\
   ```

2. When executing the upgrade script, you must specify the absolute path of the META-INF directory. The files are read from the `..\META-INF\application-sync` directory (and its sub-directories) and will then be written, along with any changes to the xsi:schemaLocation parameter, to the `..\META-INF\data` directory. In the event that `..\META-INF\data` already exists, you are asked to rename the directory to prevent any inadvertent loss of files. Note that none of the source files are changed or moved from the `..\META-INF\application-sync` directory.

3. Run the upgrade script:

   ```
   UpgradeApplicationSyncFiles c:\bea81\user_projects\applications\
   testApp\META-INF\
   ```

4. Review the corresponding log file, `upgradeApplicationSyncFiles.log`, to verify that the upgrade was successful. The same information is displayed to the console.

Step 4: Upgrade Portal and Portlet Files

Upgrade `.portal` and `.portlet` files using the script in the following directory:

```
..\weblogic81\portal\upgrade
```

From within the upgrade directory `..\weblogic81\portal\upgrade`, run the script by typing the following at the command line:

```
upgradePortalFiles -i<source dir or source file> -o<output dir>
```

For example, Listing 3-1 illustrates the output if the `sampleportal-project` directory was copied into a directory called `C:\old` and a directory called `C:\new` were created for the transformed portal and portlet files.

Note: The `upgradePortalFiles` script looks recursively through all subfolders for `.portal` and `.portlet` files. None of the source files are changed.

Listing 3-1 Running the upgradePortalFiles Script

```
C:\weblogic81\portal\upgrade>upgradePortalFiles -iC:\old -oC:\new

C:\weblogic81\portal\upgrade>REM echo off

C:\weblogic81\portal\upgrade>C:/jdk141_03/bin/java -cp

C:/weblogic81/portal/lib/netuix/ejb/netuix_util.jar;C:/weblogic81/portal/l
ib/netuix/ejb/netuix.jar;C:/weblogic81/portal/lib/netuix/system/netuix_sys
tem.jar;C:/weblogic81/p13n/lib/p13n_system.jar;C:/weblogic81/server/lib/we
blogic.jar com.bea.netuix.migration.PortalMigration -iC:\old -oC:\new

Processing file Bookmarks.portlet

Transformed version of Bookmarks.portlet was valid.

Saving file...

Processing file CompanyProfiles.portlet

Transformed version of CompanyProfiles.portlet was valid.

Saving file...

Processing file CustomerService.portlet

Transformed version of CustomerService.portlet was valid.

Saving file...

Processing file Dictionary.portlet

Transformed version of Dictionary.portlet was valid.
```

```
Saving file...
Processing file Email.portlet
Transformed version of Email.portlet was valid.
Saving file...
Processing file GroupToDo.portlet
Transformed version of GroupToDo.portlet was valid.
Saving file...
Processing file MyNewsletters.portlet
Transformed version of MyNewsletters.portlet was valid.
Saving file...
Processing file MyToDo.portlet
Transformed version of MyToDo.portlet was valid.
Saving file...
Processing file Newsletters.portlet
Transformed version of Newsletters.portlet was valid.
Saving file...
Processing file Portfolio.portlet
Transformed version of Portfolio.portlet was valid.
Saving file...
Processing file PrimaryCampaign.portlet
Transformed version of PrimaryCampaign.portlet was valid.
Saving file...
Processing file QuickLinks.portlet
Transformed version of QuickLinks.portlet was valid.
Saving file...
Processing file Quote.portlet
Transformed version of Quote.portlet was valid.
```

```
Saving file...
Processing file ReviewNewsletters.portlet
Transformed version of ReviewNewsletters.portlet was valid.
Saving file...
Processing file SecondaryCampaign.portlet
Transformed version of SecondaryCampaign.portlet was valid.
Saving file...
Processing file WebSearch.portlet
Transformed version of WebSearch.portlet was valid.
Saving file...
Processing file WhatsHot.portlet
Transformed version of WhatsHot.portlet was valid.
Saving file...
Processing file WorldNews.portlet
Transformed version of WorldNews.portlet was valid.
Saving file...
Processing file sampleportal.portal
Transformed version of sampleportal.portal was valid.
Saving file...
PortalMigration completed successfully.
C:\weblogic81\portal\upgrade>
```

The input argument (-i) should be a directory that has portals and/or portlets in it or one of its sub-directories. It is expected that users will specify a 7.0 EBCC project directory but there is no requirement that this be so. When a directory is specified for the input argument, the tool traverses the directory and all its sub-directories looking for files with a `.portal` or `.portlet` extension. Each file found is validated against the 7.0 schema. If it passes validation, the appropriate XSLT is applied. If transformation is successful, the new document is validated against the 8.1 schema.

The file is written to the output directory as described below. The input argument might also be a specific `.portal` or `.portlet` file. In this case, only the specified file is processed as described above and written out as described below.

The output argument (-o) should be a directory. Under this directory, a directory named “portal” is created and any portals found in the input directory are written here after they are transformed, if they pass Version 8.1 validation. Under the "portal" directory a directory named "portlets" is created and all portlets found in the input directory are validated and written here after they are transformed. Portals or portlets that are transformed but fail the 8.1 validations are written to a sub-directory named “failed-verification” located under the output "portal" or "portlet" directories, depending on the file's type. None of the source files are changed or moved.

The output directory looks something like this when the file upgrade completes:

```
<output dir>
      |
      |-- portal (valid 8.1 portals files in here)
            |
            |-- failed-verification (portals that failed 8.1 validation)
            |
            |-- portlets (valid 8.1 portlets files in here)
_____________|
_____________|-- failed-verification (portlets that failed 8.1 validation)
```

Step 5: Upgrade ENTITLEMENT_RULESET Records

This upgrade step is necessary *only if you are planning to run in compatibility mode*.

Like the previous upgrade action taken with the files located under the application-sync directory, records that exist in the database must also be updated to properly reflect the new xsi:schemaLocation. Specifically, the records reside in the ENTITLEMENT_RULESET table.

Upgrade ENTITLEMENT_RULESET records in the database using the following steps:

1. Back up the 7.0 database.
2. For PointBase only: change directories to `..\bea81\weblogic81\portal\upgrade` and start the PointBase server and PointBase console.

   ```
   ..\bea81\weblogic81\common\bin\startPointBase.cmd
   -ini=pointbase\pointbase.ini
   ```

   To start the PointBase Server, you might need to pass in a parameter such as the following:

```
-port=9095
```

3. Navigate to: `..\bea81\weblogic81\portal\upgrade`.
4. Update `upgrade.properties` with your 7.0 database settings.
5. Run the following script: `upgradeEntitlementRulesets`.
6. Review the `upgradeEntitlementRulesets.log` file to verify that the upgrade was successful. The same information is displayed to the console.
7. For PointBase: shut down the PointBase Server

Step 6: Upgrading Existing Behavior Tracking Data

To preserve and reuse existing behavior tracking data in an upgraded application, perform the following steps:

1. Archive existing behavior tracking data
2. Create another schema within the database for the new release.
3. Change the connection pool settings for behavior tracking to point to the new schema.

CHAPTER 4

Upgrading 8.1 to Service Pack 2

This section provides instructions for applying service pack changes to your portal applications after you install Service Pack 2. For instructions on installing service packs, see "Installing Service Packs and Rolling Patches" documentation at:

`http://e-docs.bea.com/platform/docs81/install/update.html`

You will follow this process to complete the upgrade

- Step 1: Upgrade an Existing Domain or Create a New SP2 Domain
- Step 2: Upgrade Existing Applications and Projects
- Step 3: Redeploy the Upgraded Application

Note: Upgrading to Service Pack 2 does not require a database schema update.

Step 1: Upgrade an Existing Domain or Create a New SP2 Domain

You must do one of the following

- Upgrade your existing domain to use the new SP2 libraries.
- Create a new domain with the SP2 Configuration Wizard that mirrors your existing domain.

Upgrading an Existing Domain

If you choose to upgrade your existing domain to use the new SP2 libraries, upgrade instructions vary according to the method you used to install SP2. Table 4 -2 explains the options.

Table 4 -2 GA-to-SP2 Upgrade Options

If you used...	Then...
The upgrade installer to update your existing install	You do not need to make any changes to your SP2 domain. All scripts should pick up the appropriate libraries in `/weblogic81`.

Table 4 -2 GA-to-SP2 Upgrade Options

The full installer and installed SP2 in a separate directory	You must manually update scripts in your domain. Update the following environment variables for all scripts in your domain:
	`BEA_HOME`
	`DOMAIN_HOME`
	`POINTBASE_HOME`
	`PORTAL_HOME`
	`JAVA_HOME`
	`JAVA_VENDOR`
	`JDK_HOME`
	`JDK_TOOLS`
	`WEBLOGIC_HOME`
	`WL_HOME`
	Below is a list of files in your domain to update:
	`create_db`
	`installService.cmd`
	`set-dbenv`
	`setDomainEnv`
	`setDomainEnvQS`
	`startManagedWebLogic`
	`startManagedWebLogicQS`
	`startPointBaseConsole`
	`startPointBaseConsoleQS`
	`startWebLogic`
	`startWebLogicQS`
	`stopManagedWebLogic`
	`stopManagedWebLogicQS`
	`stopWebLogic`
	`stopWebLogicQS`
	`uninstallService.cmd`
	`webappCompile`
	`webappCompileQS`
	Note: `*QS` scripts are for only the out-of-the-box domains. A domain created with the Configuration Wizard does not include these files.

If you use WebLogic Integration's Smart Update, it automatically modifies environment variables when SP2 is installed in a different directory. If you use the SP2 upgrade installer, you don't need to update any domain files, as these should already point to the new SP2 libraries.

You might need to update paths to point to the new JDK for SP2. These are:

```
JAVA_HOME
JAVA_VENDOR
JDK_HOME
JDK_TOOLS
```

These paths are not picked up automatically after running the SP2 upgrade installer because they reside in a unique location.

Creating a New SP2 Domain

To create a new SP2 domain that mirrors your existing domain, use the Configuration Wizard. See "Creating a New WebLogic Domain" at: http://edocs.bea.com/platform/docs81/confgwiz/newdom.html.

To apply the service pack library updates to portal applications already deployed in production, redeploy those applications after you have updated them in WebLogic Workshop. For deployment instructions, see "Preparing and Deploying the EAR File" at:

http://edocs.bea.com/wlp/docs81/prodOps/deployment.html.

Step 2: Upgrade Existing Applications and Projects

After you install a new service pack that includes portal library updates, you must update the libraries in the applications you have developed. Updating overwrites the existing libraries.

Before You Begin – About UUP

If you have developed your own Unified User Profile (UUP) to access user profile properties stored in an external user store, you have most likely modified and re-created the `p13n_ejb.jar` file in your application root directory. Because `p13n_ejb.jar` is one of the files overwritten in the following procedure, you should back up your existing file. After the upgrade procedure, you must re-create the updated `p13n_ejb.jar` with your UUP implementation. For more information, see "Setting up Unified User Profiles" in the *User Management Guide* at http://e-docs.bea.com/wlp/docs81/users/uup.html#999527.

Update the Portal Libraries

To update your application libraries:

1. Shut down your server if it is running; in WebLogic Workshop, choose **Tools→WebLogic Server→Stop WebLogic Server**.

2. In WebLogic Workshop, open the portal application that you want to update.
3. In the Application window, right-click the application directory and choose **Install→Update Portal Libraries**.
4. If your application uses Commerce or Pipeline components, right-click the application directory and choose **Install→Commerce Services** and **Install→Pipeline Services**.
5. After the portal application libraries are updated, a dialog box appears that lets you select Web projects in the application to update. Select the Web projects whose libraries you want to update, and click **OK**.

   If you choose not to use the dialog box to update a Web project's libraries, you can update the Web project later by right-clicking the Web project directory in the Application window and choosing **Install→Update Portal Libraries**.
6. If your application uses Commerce or Webflow JSP tag libraries, right-click the Web project directory in the Application window and choose **Install→Commerce Taglibs** and **Install→Webflow Taglibs**.
7. If you have hidden any Web applications in the WebLogic Workshop interface, those Web applications will not be updated. Either un-hide them and perform the update as described in the previous steps, or manually replace the updated libraries in the hidden Web application(s).

   To apply the service pack library updates to portal applications that are already deployed in production, redeploy those applications after you have updated them in WebLogic Workshop. See "Step 3: Redeploy the Upgraded Application."
8. If you developed your own Unified User Profile (UUP) by modifying `p13n_ejb.jar` in your application root directory, re-implement your UUP in the new `p13n_ejb.jar` file. See "Before You Begin – About UUP" on page 4-4.
9. Restart your server.

Step 3: Redeploy the Upgraded Application

The final upgrade step is to redeploy the application on your server. For deployment instructions, see "Deploying Portal Applications" at

http://edocs.bea.com/wlp/docs81/prodOps/deployment/index.html.

CHAPTER 5

Upgrading to Service Pack 3

This section provides instructions for applying Service Pack 3 (SP3) changes to your portal applications after you install that service pack. For instructions on installing service packs, see "Installing Service Packs and Rolling Patches" at:

`http://e-docs.bea.com/platform/docs81/install/update.html`

This section contains information on the following subjects:

- The Service Pack Upgrade Process
- Step 1: Upgrade an Existing SP2 Domain or Create A New SP3 Domain
- Step 2: Upgrade Existing Database Schema
- Step 3: Upgrade Existing Applications
- Step 4: Redeploy the Upgraded Application

The Service Pack Upgrade Process

Regardless of the release—8.1 or 8.1 with Service Pack 2—from which you are upgrading, you follow the same basic process to complete the upgrade.

1. Upgrade an existing SP2 domain or create a new SP3 domain.
2. Upgrade existing database schema.
3. Upgrade existing applications.

4. Redeploy upgraded application to your production environment.

The following sections describe these steps.

Step 1: Upgrade an Existing SP2 Domain or Create A New SP3 Domain

You must choose one of the following upgrade paths:

- Upgrade your existing SP2 domain to use the new SP3 libraries and database modules.

  Typically you would use this upgrade path if you want to preserve a highly customized domain, including LDAP setup.

- Create a new domain with the SP3 Configuration Wizard that mirrors your existing SP2 domain.

  Typically you would use this path if you are comfortable recreating your domain configuration, and you want to ensure that your domain is based on the latest domain template. If you are using this method, skip to Creating a New SP3 Domain.

Upgrading an Existing SP2 Domain

If you choose to upgrade your existing SP2 domain to use the new SP3 libraries, upgrade instructions vary according to the method you used to install SP3. The following sections explain your options and the required manual update steps.

WebLogic Portal Upgrade Installer

If you used the WebLogic Portal SP3 upgrade installer to update your existing SP2 install, then you do not need to make any changes to your SP2 domain. All scripts should pick up the appropriate libraries in `/weblogic81`.

Continue to "Step 2: Upgrade Existing Database Schema" on page 5-5.

WebLogic Portal Full Installer

If you used the full installer and you installed SP3 in a separate directory, then you must manually update scripts in their SP2 domain.

Update setDomainEnv

In the script `setDomainEnv`, update the following variables:

```
JAVA_HOME
BEA_JAVA_HOME
SUN_JAVA_HOME
WL_HOME
```

Note: The file `setDomainEnv` contains the `DOMAIN_NAME` variable. Do not change the value of this variable if you are upgrading an existing domain to SP4.

Update Path to JDK as Needed

The upgrade installer cannot update paths to the JDK because it can reside in a unique location. To update paths to point to the new JDK for SP4, edit the variables in the scripts shown in Table 5 -3:

Table 5 -3 Variables/Scripts Containing Path

Variable	Script
JDK_HOME BEAHOME WEBLOGIC_HOME PORTAL_HOME POINTBASE_HOME	set-dbenv
JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JDK_HOME	setDomainEnv

Review and Update Other Scripts and Variables as Needed

The following table lists other variables and files that may or may not need updating, depending on which scripts you use in your environment; however, BEA recommends that you review the files in directory *WL_HOME*\samples\domains\portal to verify you have made all needed changes.

Variable	Script/File	Notes
BEAHOME WL_HOME JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JDK_HOME POINTBASE_HOME	webappCompile startPointBaseConsole startWebLogic[1] stopManagedWebLogic stopWebLogic	[1]The file startWebLogic contains the DOMAIN_NAME variable. Do not change the value of this variable if you are upgrading an existing domain to SP3.

Adding Module Information to Database Properties File

Navigate to the file `db_settings.properties` in the domain that you are upgrading and update the module settings for SP3, as appropriate. The portal module `cmv` and netuix module `wsrp` are new for SP3. For example, the corresponding lines of the properties file should appear like this:

```
p13n_modules=p13n ds au bt
portal_modules=cm cmv wlcs wps sample_cm collaboration
netuix_modules=pf wsrp
```

If you use the script `create_db`, update these variables in the script `set-dbenv`:

```
BEA_HOME
PORTAL_HOME
JDK_HOME
JDK_TOOLS
WEBLOGIC_HOME
```

Special Considerations for LdapPropertyManager UUP in p13n_ejb.jar:

A new environment entry (env-entry) called `useSSL` has been added for SP3. In order to use `LdapPropertyManagerImpl` in WebLogic Portal SP3, you must manually change the `ejb-jar.xml` deployment to include your connection information.

If you attempt to use the old `ejb-jar.xml` deployment descriptor, the missing `useSSL` env-entry causes the following exception to occur when you try to get user properties from LDAP:

```
javax.naming.NameNotFoundException: While trying to look up useSSL in
/app/ejb/p13n_ejb.jar#LdapPropertyManager/comp/env/config.; remaining name
'useSSL'
```

If you are continuing to an SP4 upgrade after completing the steps in this section, be sure to perform this task before using SP4.

Creating a New SP3 Domain

To create a new SP3 domain that mirrors your existing SP2 domain, follow these steps:

1. Use the Configuration Wizard.

   See “Creating a New WebLogic Domain” at:
   http://edocs.bea.com/platform/docs81/confgwiz/newdom.html.

2. Copy the LDAP information from your SP2 domain to the new SP3 domain. LDAP information is located in the path: *`portal_domain`*`\`*`portal_server`*`\ldap`.

3. Re-generate any encrypted passwords that your application uses, such as passwords for database access, third-party applications or other utilities that require a login. This step is only necessary if you were already using encrypted passwords before upgrading, see "Re-Encrypting Password" on page 5-10.

Step 2: Upgrade Existing Database Schema

Scripts for upgrading an SP2 database to SP3 are located in *SP3_WL_HOME*\portal\upgrade\SP3 *where SP3_WL_HOME* is the *WL_HOME* directory for SP3.

For databases other than PointBase, in order to use the provided WebLogic Portal upgrade scripts, database client software must be installed and configured on the machine from which you execute the upgrade scripts.

Complete the steps in the following sections, as appropriate depending on your database type, to make your WebLogic Platform 8.1 SP2 database schema 8.1 SP3-compliant.

Note: For details about the database and schema changes that will be made for SP3, see "Database Changes for SP3" on page D-1.

Upgrading a PointBase Database

1. Shut down the WebLogic Server if it is running. The WebLogic Server must be shut down while performing the database schema upgrade steps.
2. Back up your existing database using a method prescribed by PointBase. Ensure that the database backup you create is valid for restore purposes if you encounter a failure during the database upgrade process.
3. Copy the `workshop.dbn`, `workshop$1.wal`, and `pointbase.ini` Pointbase database files from your source domain to *SP3_WL_HOME*\portal\upgrade\SP3.

Warning: The SP3 PointBase database upgrade scripts cannot expand `_LABEL` column lengths because PointBase does not support this type of `ALTER` function. `BOOK_LABEL`, `PAGE_LABEL`, and `LOOK_FEEL_LABEL` columns are expanded from Varchar(40) to Varchar(80). PointBase databases created with SP3 will properly define the columns as Varchar(80).

4. Make the following change to the *SP3_WL_HOME*\portal\upgrade\SP4\db_settings.properties file:
   - Set the `server`, `dblogin`, and `dbpassword` in the uncommented section for your environment, so that they correspond to the values for your existing database.

5. Navigate to *SP3_WL_HOME*\portal\upgrade\SP3 and run the upgrade_db_schema_to_81SP3.cmd or .sh script to upgrade your database schema.
6. To verify that the upgrade was successful, inspect the upgrade_db_schema_to_81SP3.log file. You can find the scripts and log file in *SP3_WL_HOME*\portal\upgrade\SP3.
7. Copy the workshop.dbn, workshop$1.wal, and pointbase.ini files from *SP3_WL_HOME*\portal\upgrade\SP3 to your SP3 domain, as appropriate.

The database is now upgraded. Back up the upgraded database before upgrading existing applications.

If you are continuing to an SP4 upgrade, be sure to back up the upgraded SP3 database before performing the SP4 upgrade.

Remove OLD_ Tables After Upgrade (Optional)

During an upgrade from WebLogic Platform 8.1 SP2 to SP3, some tables are renamed with an old_ prefix so that the upgrades can be performed successfully and data migrated to the newly created tables. Once you have successfully upgraded, you can manually drop the following tables with the old_ prefix from your schema:

- OLD_WLCS_SECURITY
- OLD_WLCS_SAVED_ITEM_LIST

Upgrading a Sybase Database

1. Shut down the WebLogic server if it is running. The WebLogic Server must be shut down while performing the database schema upgrade steps.
2. Back up your existing database using a method prescribed by Sybase. Ensure that the database backup you create is valid for restore purposes if you encounter a failure during the database upgrade process.
3. Set the "Allow Select Into/Bulkcopy" database parameter to true for the WEBLOGIC Portal database; see Listing 5-1 for an example:

Listing 5-1 Setting Allow Select Into/Bulkcopy Option in Sybase

```
use master
go
```

```
sp_dboption WEBLOGIC,"select into/bulkcopy",true
go
use WEBLOGIC
go
checkpoint
go
```

4. Make the following changes to the *SP3_WL_HOME*\portal\upgrade\SP3\db_settings.properties file:
   - Uncomment the section of the file that corresponds to Sybase
   - Set the server, dblogin, and dbpassword in the uncommented section for your environment, so that they correspond to the values for your existing database.
5. Navigate to *SP3_WL_HOME*\portal\upgrade\SP3 and run the upgrade_db_schema_to_81SP3.cmd or .sh script to upgrade your database schema to SP3, as appropriate. To verify that the upgrade was successful, inspect the upgrade_db_schema_to_81SP3.log file. You can find the scripts and log file in *SP3_WL_HOME*\portal\upgrade\SP3.

   The database is now upgraded. Back up the upgraded database before upgrading existing applications.
6. Reset the "Allow Select Into/Bulkcopy" database parameter to false for the WEBLOGIC Portal database, as shown in Listing 5-2.

Listing 5-2 Resetting "Allow Select into/Bulkcopy" Option in Sybase

```
use master
go
sp_dboption WEBLOGIC,"select into/bulkcopy",false
go
use WEBLOGIC
go
checkpoint
go
```

If you are continuing to an SP4 upgrade, be sure to back up the upgraded SP3 database before performing the SP4 upgrade.

Remove OLD_ Tables After Upgrade (Optional)

During an upgrade from WebLogic Platform 8.1 SP2 to SP3, some tables are renamed with an `old_` prefix so that the upgrades can be performed successfully and data migrated to the newly created tables. Once you have successfully upgraded, you can manually drop the following tables with the `old_` prefix from your schema:

- `OLD_WLCS_SECURITY`
- `OLD_WLCS_SAVED_ITEM_LIST`

Upgrading a SQL Server Database

1. Shut down the WebLogic Server if it is running. The WebLogic Server must be shut down while performing the database schema upgrade steps.
2. Back up your existing database using a method prescribed by SQL Server. Ensure that the database backup you create is valid for restore purposes if you encounter a failure during the database upgrade process.
3. Make the following changes to the *SP3_WL_HOME*`\portal\upgrade\SP3\db_settings.properties` file:
   - Uncomment the section of the file that corresponds to SQL Server
   - Set the `server`, `dblogin`, and `dbpassword` in the uncommented section for your environment, so that they correspond to the values for your existing database.
4. Navigate to *SP3_WL_HOME*`\portal\upgrade\SP3` and run the `upgrade_db_schema_to_81SP3.cmd` or `.sh` script to upgrade your database schema. To verify that the upgrade was successful, inspect the `upgrade_db_schema_to_81SP3.log` file. You can find the scripts and log file in *SP3_WL_HOME*`\portal\upgrade\SP3`.

   The database is now upgraded. Back up the upgraded database before upgrading existing applications.

If you are continuing to an SP4 upgrade, be sure to back up the upgraded SP3 database before performing the SP4 upgrade.

Remove OLD_ Tables After Upgrade (Optional)

During an upgrade from WebLogic Platform 8.1 SP2 to SP3, some tables are renamed with an `old_` prefix so that the upgrades can be performed successfully and data migrated to the newly created tables. Once you have successfully upgraded, you can manually drop the following tables with the `old_` prefix from your schema:

- `OLD_WLCS_SECURITY`
- `OLD_WLCS_SAVED_ITEM_LIST`

Upgrading an Oracle or DB2 Database

1. Shut down the WebLogic Server if it is running. The WebLogic Server must be shut down while performing the database schema upgrade steps.
2. Back up your existing database using a method prescribed by your database vendor. Ensure that the database backup you create is valid for restore purposes if you encounter a failure during the database upgrade process.
3. Make the following changes to the *SP3_WL_HOME*`\portal\upgrade\SP3\db_settings.properties` file:
   - Uncomment the section of the file that corresponds to Oracle or DB2, as appropriate
   - Set the `server`, `dblogin`, and `dbpassword` in the uncommented section for your environment, so that they correspond to the values for your existing database.
4. Navigate to *SP3_WL_HOME*`\portal\upgrade\SP3` and run the `upgrade_db_schema_to_81SP3.cmd` or `.sh` script to upgrade your database schema. To verify that the upgrade was successful, inspect the `upgrade_db_schema_to_81SP3.log` file. You can find the scripts and log file in *SP3_WL_HOME*`\portal\upgrade\SP3`.

   The database is now upgraded. Back up the upgraded database before upgrading existing applications.

If you are continuing to an SP4 upgrade, be sure to back up the upgraded SP3 database before performing the SP4 upgrade.

Remove OLD_ Tables After Upgrade (Optional)

During an upgrade from WebLogic Platform 8.1 SP2 to SP3, some tables are renamed with an `old_` prefix so that the upgrades can be performed successfully and data migrated to the newly created tables. Once you have successfully upgraded, you can manually drop the following tables with the `old_` prefix from your schema:

- OLD_WLCS_SECURITY
- OLD_WLCS_SAVED_ITEM_LIST

Step 3: Upgrade Existing Applications

When you upgrade, you may need to also update applications you have developed. Specifically, additional tasks are required if the following is true:

- You are also switching to a different domain when you upgrade, AND you use encrypted passwords in your application configuration, you'll need to re-encrypt those passwords.
- If you are installing a new service pack that includes portal library updates, you must update the libraries in the applications you have developed. Updating overwrites the existing libraries.

Re-Encrypting Password

If you created a new domain during your upgrade process AND your application uses encrypted passwords, you must re-encrypt any passwords currently used by your application.

A portal domain's portal application's META-INF directory contains the application-config.xml file. In newly created portal domains where the portal application is deployed, this file will contain unencrypted passwords. These passwords actually remain clear text until the portal administration tool's Service Administration page is used to edit and save attributes contained in this config file, or the application using the password accesses it. (And if the portal application is deployed as an ear file, this file cannot be edited and saved.)

<TONIA> Encrypting passwords is done differently according to how your application is deployed. If you have deployed your application as an EAR file, you'll have to manually modify the configuration files and bring down the server to do so. If you have not deployed as an EAR, you can use the Service Administration tools within the Administration Console to enter the passwords for your application. The Service Administration tools then automatically encrypt your passwords.

<TONIA> Is the application already deployed at this point? Other than WSRP and CM SPI implementations, what other passwords are kept in the application-config file?

<TONIA> Is there a way to automatically encrypt passwords for CM SPI implementations? Or is this manual process always necessary when switching domains?

You'll need to use the `EncryptDomainString` command-line utility to generate an encrypted password and then place that encrypted password into your portal applications's

application-config.xml file while it is still in the development environment. Then you can build the EAR file for the application and deploy it as necessary.

Encrypting Passwords

To encrypt passwords, use this procedure:

1. Open a command box (DOS shell) and navigate to *domain*/portal/ (where *domain* is the domain directory for the application) and run setDomainEnv.cmd.
2. At the prompt, enter

   ```
   java com.bea.p13n.util.EncryptDomainString -targetDomainDir d
   -inputString s
   ```

   where:

   – *d* is the domain directory to which the portal application is being deployed;

   – *s* is the input password to encrypt

   For example:

   ```
   java com.bea.p13n.util.EncryptDomainString -targetDomainDir
   \bea\weblogic81b\samples\domain\portal -inputString weblogic
   ```

   In this example, the input string weblogic represents administrator's password (adminpassword=weblogic; see Listing 5-3). The command line utility prints a domain specific encrypted string.
3. Open WebLogic Workshop and the specific portal application.
4. Select File>Open>File (Figure 5-1).

Figure 5-1 Opening a File in WebLogic Workshop

An Open dialog box appears.

5. Navigate to the application's META-INF folder and open application-config.xml.

Listing 5-3 Clear text Passwords in application-config.xml

```
<ConsumerSecurity AdminPassword="weblogic" AdminUserName="weblogic"
   CertAlias="wsrpConsumer" CertPrivateKeyPassword="wsrppassword"
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword="password"
   Name="ConsumerSecurity"/>
```

6. Replace the clear text passwords with those generated by the `EncryptDomainString` utility, as shown in Figure 5-4. You will need to run `EncryptDomainString` for each password in the `<ConsumerSecurity>` (Listing 5-3) element; for example:
   - `AdminPassword="weblogic"`
   - `CertPrivateKeyPassword="wsrppassword"`
   - `KeystorePassword="password"`

Listing 5-4 Encrypted Passwords Generated by EncryptDomainString Utility

```
<ConsumerSecurity AdminPassword="{3DES}3QrrUeIwN/DxlDI++1ixPw=="
   AdminUserName="weblogicc" CertAlias="wsrpConsumer"
   CertPrivateKeyPassword="{3DES}g7h+VOSAsO9pSlvYSSB2iw=="
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword=
      "{3DES}1OLYVirMWOo+3sEU80cMqw=="

   Name="ConsumerSecurity" />
```

Now you can build the EAR file and deploy the application.

Note on Changing Passwords

If you need to change an administrator's password for any reason, simply changing the password will result in having to rebuild and redeploy the EAR. This is time-consuming and counterproductive. Instead, you can work around this problem by doing the following:

1. Create a special user in the target system for a WSRP administrator. See Create a New User for more information on creating a user.
2. Make that user a member of the administrator group. See Add a User to a Group for more information on adding a member to a group.
3. Insert the new logon information (username and password) into the application's `application-config.xml` file as described in Encrypting Passwords.

Before You Begin – About UUP

If you have developed your own Unified User Profile (UUP) to access user profile properties stored in an external user store, you have most likely modified and re-created the `p13n_ejb.jar` file in your application root directory. Because `p13n_ejb.jar` is one of the files overwritten in the following procedure, you should back up your existing file. After the upgrade procedure, you must re-create the updated `p13n_ejb.jar` with your UUP implementation. For more information, see "Setting up Unified User Profiles" in the *User Management Guide* at http://e-docs.bea.com/wlp/docs81/users/uup.html#999527.

Update the Portal Libraries

To update your application libraries, follow these steps:

1. Ensure that WebLogic Workshop 8.1 SP2 is running. If it isn't, start it.
2. If your WebLogic server is running, shut it down by choosing, in WebLogic Workshop, **Tools→WebLogic Server→Stop WebLogic Server**.
3. In WebLogic Workshop, open the portal application that you want to update.
4. In the Application window, right-click the application directory and choose **Install→Update Portal Libraries**.
5. After the portal application libraries are updated, a dialog box appears that lets you select Web projects in the application to update. Select the Web projects whose libraries you want to update, and click **OK**.

   If you choose not to update a Web project's libraries with the dialog box, you can update the Web project later by right-clicking the Web project directory in the Application window and choosing **Install→Update Portal Libraries**.
6. If your application uses Commerce or Pipeline components, right-click the application directory and choose **Install→Commerce Services** and **Install→Pipeline Services**.

7. If your application uses Commerce or Webflow JSP tag libraries, right-click the Web project directory in the Application window and choose **Install→Commerce Taglibs** and **Install→Webflow Taglibs**.
8. If you have hidden any Web applications in the WebLogic Workshop interface, those Web applications will not be updated. Either un-hide them and perform the update as described in the previous steps, or manually replace the updated libraries in the hidden Web application(s).

   To apply the service pack library updates to portal applications already deployed in production, redeploy those applications after you have updated them in WebLogic Workshop. See "Step 4: Redeploy the Upgraded Application" for deployment instructions.
9. If you developed your own Unified User Profile (UUP) by modifying `p13n_ejb.jar` in your application root directory, re-implement your UUP in the new `p13n_ejb.jar` file. See "Before You Begin – About UUP" on page 5-13.
10. Restart the server.

Step 4: Redeploy the Upgraded Application

The final upgrade step is to redeploy the application on your server.

For deployment instructions, see "Preparing and Deploying the EAR File" at:

http://edocs.bea.com/wlp/docs81/prodOps/deployment.html.

CHAPTER 6

Upgrading to Service Pack 4

This section provides instructions for applying Service Pack 4 (SP4) changes to your portal applications after you install that service pack. For instructions on installing service packs, see "Installing Service Packs and Rolling Patches" at:

`http://e-docs.bea.com/platform/docs81/install/update.html`

This section contains information on the following subjects:

- The Service Pack Upgrade Process
- Step 1: Upgrade an Existing SP2 or SP3 Domain or Create A New SP4 Domain
- Step 2: Upgrade Existing Database Schema
- Step 3: Upgrade Existing Applications
- Step 4: Redeploy the Upgraded Application
- Step 5: Review Functional Changes for SP4

For detailed information on the changes made for SP4, see the following sections:

- Functional Changes Affecting Your WebLogic Portal Environment
- Changes to Visitor Tools for SP4
- Database Changes for SP4

The Service Pack Upgrade Process

Regardless of the release—8.1 or 8.1 with a previous service pack—from which you are upgrading, you follow the same basic process to complete the upgrade.

1. Upgrade an existing SP2 or SP3 domain, or create a new SP4 domain.
2. Upgrade existing database schema.
3. Upgrade existing applications.
4. Redeploy upgraded application to your production environment.
5. Perform manual upgrade tasks to enable SP4 enhancements.

The following sections describe these steps.

Step 1: Upgrade an Existing SP2 or SP3 Domain or Create A New SP4 Domain

You must choose one of the following upgrade paths:

- Upgrade your existing SP2 or SP3 domain to use the new SP4 libraries and database modules.

  Typically you would use this upgrade path if you want to preserve a highly customized domain, including LDAP setup.

- Create a new domain with the SP4 Configuration Wizard that mirrors your existing SP2 or SP3 domain.

  Typically you would use this path if you are comfortable recreating your domain configuration, and you want to ensure that your domain is based on the latest domain template. If you are using this method, skip to "Creating a New SP4 Domain" on page 6-6.

Upgrading an Existing SP2 or SP3 domain

If you choose to upgrade your existing SP2 or SP3 domain to use the new SP4 libraries, upgrade instructions vary according to the method you used to install SP4. The following sections explain your options and the required manual update steps.

WebLogic Portal Upgrade Installer

If you used the WebLogic Portal SP4 upgrade installer to update your existing SP2 or SP3 install, then you do not need to make any changes to your SP2 or SP3 domain. All scripts should pick up the appropriate new libraries for the SP4 environment.

Continue to "Step 2: Upgrade Existing Database Schema" on page 6-6.

Update Paths to JDK as Needed

The upgrade installer cannot update paths to the JDK because it can reside in a unique location. To update paths to point to the new JDK for SP4, edit the variables in the scripts shown in Table 6 -4:

Table 6 -4 Variables/Scripts Containing Path

Variable	Script
JDK_HOME BEAHOME WEBLOGIC_HOME PORTAL_HOME POINTBASE_HOME	set-dbenv
JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JDK_HOME	setDomainEnv

Edit URL Template File for Use with WSRP

The `url-template-config.xml` file, which is automatically created in a portal Web project, contains URL templates and variables for WSRP portlets. You must make the following two changes in this file for each WSRP URL template, so that the BEA-provided consumer will be able to honor mode changes and state changes from non-BEA producers:

- Replace `_mode` with `wsrp-mode`
- Replace `_state` with `wsrp-windowState`

In addition, the producer should use the `wsrp:` prefix for standard modes and window states in URLs, so that non-BEA consumers can recognize URLs returned by BEA-provided producers.

WebLogic Portal Full Installer

If you used the full installer and you installed SP4 in a separate directory, then you must manually update scripts in your SP2 or SP3 domain.

Update setDomainEnv

In the script `setDomainEnv`, update the following variables as needed:

```
JAVA_HOME
BEA_JAVA_HOME
SUN_JAVA_HOME
WL_HOME
```

Note: The file `setDomainEnv` contains the `DOMAIN_NAME` variable. Do not change the value of this variable if you are upgrading an existing domain to SP4.

Update Path to JDK as Needed

You might need to update the path to point to the new JDK for SP4. The variables and their corresponding scripts are shown in Table 6 -5:

Table 6 -5 Variables/Scripts Containing Path

Variable	Script/File
JDK_HOME JDK_TOOLS	set-dbenv
JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JAVA_VENDOR JDK_HOME	setDomainEnv

Review and Update Other Scripts and Variables as Needed

The following table lists other variables and files that may or may not need updating, depending on which scripts you use in your environment; however, BEA recommends that you review the files in your existing domain directory to verify that you have made all needed changes.

Table 6 -6 Other Variables and Files

Variable	Script/File	Notes
BEAHOME WL_HOME JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JDK_HOME POINTBASE_HOME	webappCompile startPointBaseConsole startWebLogic[1] stopManagedWebLogic stopWebLogic	[1]The file startWebLogic contains the DOMAIN_NAME variable. Do not change the value of this variable if you are upgrading an existing domain to SP4.
wlsHome.path jdkHome.path	workshop.properties	

Add Module Information to Database Properties File

If you are planning to reinitialize your database using the `create_db.cmd/.sh` script, you must manually update the `db_settings.properties` file, in the domain that you are upgrading, to add new module information. Later you will update the `db_settings.properties` file with database-specific information.

Navigate to the file `db_settings.properties` in the domain that you are upgrading and edit the module settings for SP4, as the following example shows. The portal module `cmv` and netuix module `wsrp` were added for SP3; the p13n module `er` is new in SP4. For more information on these new modules, see "Database Changes for SP3" on page D-1 and "Database Changes for SP4" on page C-1.

The corresponding lines of the properties file should appear like this:

```
p13n_modules=p13n ds au bt er
portal_modules=cm cmv wlcs wps sample_cm collaboration
netuix_modules=pf wsrp
```

Update set-dbenv if You Plan to Run create_db

If you use the script `create_db`, update these variables in the script `set-dbenv`:

```
BEA_HOME
PORTAL_HOME
JDK_HOME
JDK_TOOLS
WEBLOGIC_HOME
```

Back Up LDAP Data

In Service Pack 4, policy reference data is stored in the RDBMS rather than in LDAP. When you access the Administration Portal for the first time in an upgraded domain, an automatic migration of this policy reference data occurs.

Back up your LDAP data prior to continuing with the upgrade process. You can make a backup by simply copying the *portal_domain*\portalServer\ldap directory.

Creating a New SP4 Domain

To create a new SP4 domain that mirrors your existing SP2 or SP3 domain, follow these steps:

1. Use the Configuration Wizard.

   See "Creating a New WebLogic Domain" at: http://edocs.bea.com/platform/docs81/confgwiz/newdom.html.

2. Copy the LDAP information from your SP2 or SP3 domain to the new SP4 domain. LDAP information is located in the path: *portal_domain*\portalServer\ldap.

Step 2: Upgrade Existing Database Schema

Warning: If you are upgrading from an SP2 database to an SP4 database, you must first follow the steps to upgrade your database from SP2 to SP3 as described in "Step 2: Upgrade Existing Database Schema" on page 5-5. After following those steps, you can then follow the steps in this chapter to upgrade your database to SP4.

Scripts for upgrading an SP3 database to SP4 are located in *SP4_WL_HOME*\portal\upgrade\SP4 *where SP4_WL_HOME* is the *WL_HOME* directory for SP4.

For databases other than PointBase, in order to use the provided WebLogic Portal upgrade scripts, database client software must be installed and configured on the machine from which you execute the upgrade scripts.

Complete the steps in the following sections, as appropriate depending on your database type, to make your WebLogic Platform 8.1 SP3 database schema 8.1 SP4-compliant.

Upgrading a PointBase Database

1. Shut down the WebLogic Server if it is running. The WebLogic Server must be shut down while performing the database schema upgrade steps.

2. Back up your existing database using a method prescribed by PointBase. Ensure that the database backup you create is valid for restore purposes if you encounter a failure during the database upgrade process.
3. Copy the `workshop.dbn`, `workshop$1.wal`, and `pointbase.ini` Pointbase database files from your source domain to *`SP4_WL_HOME`*`\portal\upgrade\SP4`.
4. Make the following change to the *`SP4_WL_HOME`*`\portal\upgrade\SP4\db_settings.properties` file:
   - Set the `server`, `dblogin`, and `dbpassword` in the uncommented section for your environment, so that they correspond to the values for your existing database.
5. Run the `upgrade_db_schema_to_81SP4.cmd` or `.sh` script to upgrade your database schema.
6. To verify that the upgrade was successful, inspect the `upgrade_db_schema_to_81SP4.log` file. You can find the scripts and log file in *`SP4_WL_HOME`*`\portal\upgrade\SP4`.

   The following errors will appear in the `.log` file and can be ignored:

```
SQL> create user weblogic password weblogic;
SQL>
SQL> Error Message: You must have DBA-level authority to perform this
operation.
SQL> Error SQLState: 2D086
SQL> Error Code:     15088
SQL>
SQL>
create schema weblogic authorization weblogic
SQL>
SQL> Error Message:  Schema WEBLOGIC already exists in the database.
SQL> Error SQLState: 2201F
SQL> Error Code:     25228
SQL>
SQL> --
```

7. Copy the `workshop.dbn`, `workshop$1.wal`, and `pointbase.ini` files from *`SP4_WL_HOME`*`\portal\upgrade\SP4` to your SP4 domain, as appropriate.

   Note: This step is applicable whether you are using an upgraded domain or a new domain.

The database is now upgraded. Back up the upgraded database before upgrading existing applications.

Upgrading a Sybase Database

1. Shut down the WebLogic Server if it is running. The WebLogic Server must be shut down while performing the database schema upgrade steps.
2. Back up your existing database using a method prescribed by Sybase. Ensure that the database backup you create is valid for restore purposes if you encounter a failure during the database upgrade process.
3. Set the "Allow Select Into/Bulkcopy" database parameter to `true` for your `WEBLOGIC` Portal database; see Listing 6-1 for an example:

Listing 6-1 Setting Allow Select Into/Bulkcopy Option in Sybase

```
use master
go
sp_dboption WEBLOGIC,"select into/bulkcopy",true
go
use WEBLOGIC
go
checkpoint
go
```

4. Add a `WEBLOGIC_INDEX SEGMENT` to your `WEBLOGIC` Portal database for placement of non-clustered indexes in their own segment; see Listing 6-2 for an example:

Listing 6-2 Adding a WEBLOGIC_INDEX SEGMENT for Sybase

```
declare @vdevno int
select @vdevno = max(convert(tinyint, substring(convert(binary(4),d.low),
v.low,1))) + 1
from master.dbo.sysdevices d, master.dbo.spt_values v
where v.type = 'E' and v.number = 3
disk init name='WEBLOGIC_INDEX',
physname='F:\INDEXFILE\WEBLOGIC_INDEX1.DAT',
vdevno = @vdevno ,
```

```
size=15360
go
alter database WEBLOGIC ON WEBLOGIC_INDEX = 30
go
exec sp_addsegment 'WEBLOGIC_INDEX', 'WEBLOGIC','WEBLOGIC_INDEX'
go
exec sp_dropsegment 'logsegment', 'WEBLOGIC','WEBLOGIC_INDEX'
go
```

Note: You can use the `sp_estspace` stored procedure to determine the size necessary for WEBLOGIC_INDEX.

5. Make the following changes to the *SP4_WL_HOME*`\portal\upgrade\SP4\db_settings.properties` file:
   - Uncomment the section of the file that corresponds to Sybase
   - Set the `server`, `dblogin`, and `dbpassword` in the uncommented section for your environment, so that they correspond to the values for your existing database.
6. Navigate to *SP4_WL_HOME*`\portal\upgrade\SP4` and run the `upgrade_db_schema_to_81SP4.cmd` or `.sh` script to upgrade your database schema to SP4, as appropriate. To verify that the upgrade was successful, inspect the `upgrade_db_schema_to_81SP4.log` file. You can find the scripts and log file in *SP4_WL_HOME*`\portal\upgrade\SP4`.

   The database is now upgraded. Back up the upgraded database before upgrading existing applications.
7. Reset the "Allow Select Into/Bulkcopy" database parameter to false for the `WEBLOGIC` Portal database, as shown in Listing 6-3.

Listing 6-3 Resetting "Allow Select into/Bulkcopy" Option in Sybase

```
use master
go
sp_dboption WEBLOGIC,"select into/bulkcopy",false
go
use WEBLOGIC
go
```

```
checkpoint
go
```

Upgrading a SQL Server Database

1. Shut down the WebLogic Server if it is running. The WebLogic Server must be shut down while performing the database schema upgrade steps.
2. Back up your existing database using a method prescribed by SQL Server. Ensure that the database backup you create is valid for restore purposes if you encounter a failure during the database upgrade process.
3. Add the `WEBLOGIC_INDEX` file group to your `WEBLOGIC` Portal database for placement of non-clustered indexes in their own file group; see Listing 6-4 for an example:

Listing 6-4 Adding the WEBLOGIC_INDEX file group for SQL Server

```
use WEBLOGIC
go
ALTER DATABASE WEBLOGIC ADD FILEGROUP [WEBLOGIC_INDEX]
GO
ALTER DATABASE WEBLOGIC ADD FILE(NAME = N'WEBLOGIC_INDEX', FILENAME =
N'F:\INDEXFILE\WEBLOGIC_INDEX.NDF' , SIZE = 60, FILEGROWTH = 10%) TO
FILEGROUP [WEBLOGIC_INDEX]
GO
```

4. Make the following changes to the *SP4_WL_HOME*`\portal\upgrade\SP4\db_settings.properties` file:
   - Uncomment the section of the file that corresponds to SQL Server
   - Set the `server`, `dblogin`, and `dbpassword` in the uncommented section for your environment, so that they correspond to the values for your existing database.

5. Navigate to *SP4_WL_HOME*\portal\upgrade\SP4 and run the upgrade_db_schema_to_81SP4.cmd or .sh script to upgrade your database schema. To verify that the upgrade was successful, inspect the upgrade_db_schema_to_81SP4.log file. You can find the scripts and log file in *SP4_WL_HOME*\portal\upgrade\SP4.

   The database is now upgraded. Back up the upgraded database before upgrading existing applications.

Upgrading an Oracle or DB2 Database

1. Shut down the WebLogic Server if it is running. The WebLogic Server must be shut down while performing the database schema upgrade steps.
2. Back up your existing database using a method prescribed by your database vendor. Ensure that the database backup you create is valid for restore purposes if you encounter a failure during the database upgrade process.
3. Make the following changes to the *SP4_WL_HOME*\portal\upgrade\SP4\db_settings.properties file:
   - Uncomment the section of the file that corresponds to Oracle or DB2, as appropriate
   - Set the server, dblogin, and dbpassword in the uncommented section for your environment, so that they correspond to the values for your existing database.
4. *For DB2 only.* If you are not already using the DB2 command line processor (CLP) for this process, activate it now. You must run the upgrade script from the CLP.
5. Navigate to *SP4_WL_HOME*\portal\upgrade\SP4 and run the upgrade_db_schema_to_81SP4.cmd or .sh script to upgrade your database schema. To verify that the upgrade was successful, inspect the upgrade_db_schema_to_81SP4.log file. You can find the scripts and log file in *SP4_WL_HOME*\portal\upgrade\SP4.

   The database is now upgraded. Back up the upgraded database before upgrading existing applications.

Step 3: Upgrade Existing Applications

When you upgrade from SP3 to SP4, you must change the following configurations for your application:

- Re-configure your third-party content management system, if applicable.
- Re-set the passwords for your WSRP portlets, if applicable
- Update the portal libraries in any applications you developed. Updating overwrites the existing libraries.

Re-Configure Third-Party Content Management Systems

If you are using a third-party content management system, you'll need to re-configure that repository before deploying. This allows the passwords to be re-encrypted for the new domain. If you are not switching domains when upgrading, you may skip this task.

First, remove the existing repository connection, using the Portal Administration Tools:

1. View the **Repository List** tree by selecting Repository from the View drop-down list.
2. In the **Manage Repositories** tree, right-click the repository you want to remove.
3. Select **Remove** from the pop-up menu.

   Note: Be sure to write down all the configuration information you will need to re-enter.
4. A pop-up window displays confirming that you want to remove the repository. If you want to proceed, click `OK`.

Then, re-connect to the repository.

1. View the **Repository List** tree by selecting Repository from the View drop-down list.
2. In the **Manage Repositories** tree, right click the Virtual Content Repository.
3. Select `Add Repository` from the pop-up menu.
4. In the `Editor` pane, provide the connection information
5. In the **Cache Properties** box, edit the cache properties, if necessary. You can also choose to accept the default cache settings.
6. Click **Create**.

Re-Encrypting Passwords for WSRP Producers

If you are moving to a new domain when upgrading, you'll need to re-encrypt any passwords that are used to access WSRP portlets. Encryption is domain-specific so it needs to be updated when switching domains.

If you are deploying your application as an EAR file, this needs to be done manually with the `EncryptDomainString` command-line utility which is used to generate an encrypted password and then place that encrypted password into the `application-config.xml` file before you deploy your application. The application must already exist in the new domain.

Note: If you are going to deploy an exploded application (not an EAR), you can use the Service Administration tool in Portal Administration Portal to re-set passwords. When passwords are re-entered using the Service Administration tool, they are automatically encrypted. However, the Portal Administration Tool cannot be used if you have already compressed your application to an EAR file.

A portal application's `META-INF` directory contains the respective `application-config.xml.` file where WSRP passwords are stored automatically. For example, Listing 6-5 shows the WSRP element from `application-config.xml` during the development phase (exploded) before deployment as an EAR file:

Listing 6-5 Development Phase Clear Text Passwords in `application-config.xml`

```
<ConsumerSecurity AdminPassword="weblogic" AdminUserName="weblogic"
   CertAlias="wsrpConsumer" CertPrivateKeyPassword="wsrppassword"
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword="password"
   Name="ConsumerSecurity"/>
```

After the Service Administration tool is used to edit attributes, the file is saved and automatically passwords are encrypted, as shown in Listing 6-6:

Listing 6-6 .Encrypted Passwords in `application-config.xml`

```
<ConsumerSecurity AdminPassword="{3DES}3QrrUeIwN/DxlDI++1ixPw=="
   AdminUserName="weblogicc" CertAlias="wsrpConsumer"
```

```
CertPrivateKeyPassword="{3DES}g7h+VOSAsO9pSlvYSSB2iw=="
ConsumerName="wsrpConsumer"
IdentityAssertionProviderClass="com.bea.wsrp.security.
   DefaultIdentityAssertionProvider"
Keystore="wsrpKeystore.jks" KeystorePassword=
   "{3DES}1OLYVirMWOo+3sEU80cMqw=="

Name="ConsumerSecurity" />
```

Encrypting Passwords

When upgrading applications that are stored as EAR files, you must manually update the encryption.

To encrypt passwords, use this procedure:

1. Open a command box (DOS shell) and navigate to *domain*/portal/ (where *domain* is the domain directory for the application) and run setDomainEnv.cmd.
2. At the prompt, enter

   ```
   java com.bea.p13n.util.EncryptDomainString -targetDomainDir d
   -inputString s
   ```

   where:

   – *d* is the domain directory to which the portal application is being deployed; for example,

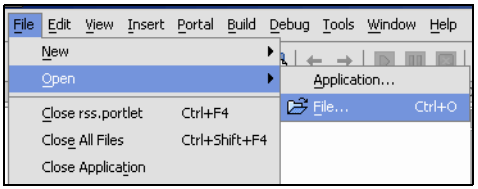
   – *s* is the input password to encrypt

   For example:

   ```
   java com.bea.p13n.util.EncryptDomainString -targetDomainDir
   \bea\weblogic81b\samples\domain\portal -inputString weblogic
   ```

   In this example, the input string weblogic represents administrator's password (adminpassword=weblogic; see Listing 6-7). The command line utility prints a domain specific encrypted string.
3. Open WebLogic Workshop and the specific portal application.
4. Select File>Open>File (Figure 6-1).

Figure 6-1 Opening a File in WebLogic Workshop



An Open dialog box appears.

5. Navigate to the application's `META-INF` folder and open `application-config.xml`.

Listing 6-7 Clear text Passwords in application-config.xml

```
<ConsumerSecurity AdminPassword="weblogic" AdminUserName="weblogic"
   CertAlias="wsrpConsumer" CertPrivateKeyPassword="wsrppassword"
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword="password"
   Name="ConsumerSecurity"/>
```

6. Replace the previously encrypted passwords with those generated by the `EncryptDomainString` utility, as shown in Figure 6-8. You will need to run `EncryptDomainString` for each password in the `<ConsumerSecurity>` (Listing 6-7) element; for example:
   - `AdminPassword="weblogic"`
   - `CertPrivateKeyPassword="wsrppassword"`
   - `KeystorePassword="password"`

Listing 6-8 Encrypted Passwords Generated by EncryptDomainString Utility

```
<ConsumerSecurity AdminPassword="{3DES}3QrrUeIwN/DxlDI++1ixPw=="
   AdminUserName="weblogicc" CertAlias="wsrpConsumer"
   CertPrivateKeyPassword="{3DES}g7h+VOSAsO9pSlvYSSB2iw=="
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
```

```
Keystore="wsrpKeystore.jks" KeystorePassword=
   "{3DES}1OLYVirMWOo+3sEU80cMqw=="

Name="ConsumerSecurity" />
```

Note on Changing Passwords

If you need to change an administrator's password for any reason, simply changing the password will result in having to rebuild and redeploy the EAR. This is time-consuming and counterproductive. Instead, you can work around this problem by doing the following:

1. Create a special user in the target system for a WSRP administrator. See Create a New User for more information on creating a user.
2. Make that user a member of the administrator group. See Add a User to a Group for more information on adding a member to a group.
3. Insert the new logon information (username and password) into the application's `application-config.xml` file as described in Encrypting Passwords.

Updating Portal Libraries

After you install a new service pack that includes portal library updates, you must update the libraries in the applications you have developed. Updating overwrites the existing libraries.

Before You Begin – About UUP

If you have developed your own Unified User Profile (UUP) to access user profile properties stored in an external user store, you have most likely modified and re-created the `p13n_ejb.jar` file in your application root directory. Because `p13n_ejb.jar` is one of the files overwritten in the following procedure, you should back up your existing file. After the upgrade procedure, you must re-create the updated `p13n_ejb.jar` with your UUP implementation. For more information, see "Setting up Unified User Profiles" in the *User Management Guide* at http://e-docs.bea.com/wlp/docs81/users/uup.html#999527.

Update the Portal Libraries

To update your application libraries, follow these steps:

1. Locate and delete the `.workshop` directory from the application that you will be working with. Do this prior to starting WebLogic Workshop.

2. Start WebLogic Workshop 8.1 SP4.
3. If your WebLogic server is running, shut it down by choosing, in WebLogic Workshop, **Tools→WebLogic Server→Stop WebLogic Server**.
4. In WebLogic Workshop SP4, open the portal application that you want to update; for example, navigate to the SP3 or SP2 Portal application.
5. In the Application window, right-click the application directory and choose **Install→Update Portal Libraries**.

   After the portal application libraries are updated, a window appears; continue to the next step.
6. In the displayed window, select the Web projects whose libraries you want to update, and click **OK**. For each selected Web project, a warning window appears. Click **Yes** to continue with the update.

   If you choose not to update a Web project's libraries using the displayed window, you can update the Web project later by right-clicking the Web project directory in the Application window and choosing **Install→Update Portal Libraries**.
7. If your application uses Commerce or Pipeline components, right-click the application directory and choose **Install→Commerce Services** and **Install→Pipeline Services**.
8. If your application uses Commerce or Webflow JSP tag libraries, right-click the Web project directory in the Application window and choose **Install→Commerce Taglibs** and **Install→Webflow Taglibs**.
9. If you have hidden any Web applications in the WebLogic Workshop interface, those Web applications will not be updated. Either un-hide them and perform the update as described in the previous steps, or manually replace the updated libraries in the hidden Web application(s).

   To apply the service pack library updates to portal applications already deployed in production, redeploy those applications after you have updated them in WebLogic Workshop. See "Step 4: Redeploy the Upgraded Application" for deployment instructions.
10. Update your Portal server settings if needed by selecting Tools→WebLogic Server→Server Properties and editing this information.
11. If you developed your own Unified User Profile (UUP) by modifying `p13n_ejb.jar` in your application root directory, re-implement your UUP in the new `p13n_ejb.jar` file. See "Before You Begin – About UUP" on page 6-16.

12. Rebuild the application by selecting Build→Build Application.
13. Restart the server in the upgraded domain.

Step 4: Redeploy the Upgraded Application

The final upgrade step is to redeploy the application on your server.

For deployment instructions, see "Preparing and Deploying the EAR File" at:

http://edocs.bea.com/wlp/docs81/prodOps/deployment.html.

Step 5: Review Functional Changes for SP4

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 7

Upgrading to Service Pack 5

This section provides instructions for applying Service Pack 5 (SP5) changes to your portal applications after you install that service pack. For instructions on installing service packs, see "Installing Service Packs and Rolling Patches" at:

`http://e-docs.bea.com/platform/docs81/install/update.html`

This section contains information on the following subjects:

- Step 1: Upgrade an Existing SP3 or SP4 Domain or Create A New SP5 Domain
- Step 2: Upgrade Existing Portal Framework Data
- Step 3: Upgrade Existing Applications
- Step 4: Redeploy the Upgraded Application
- Step 5: Review Functional Changes for SP5

For detailed information on functional changes that might require you to perform manual tasks or configuration changes, see the following sections:

- Functional Changes Affecting Your WebLogic Portal Environment

The Service Pack Upgrade Process

Regardless of the release—8.1 or 8.1 with a previous service pack—from which you are upgrading, you follow the same basic process to complete the upgrade.

1. Upgrade an existing SP3 or SP4 domain, or create a new SP5 domain.

2. Upgrade existing database data.

   Note: No database schema changes are required for Service Pack 5; however, you must update the default markup in the database, using the provided script, to be SP5 compliant.

3. Upgrade existing applications.
4. Redeploy upgraded application to your production environment.
5. Perform manual upgrade tasks to enable SP5 enhancements, if applicable.

The following sections describe these steps.

Step 1: Upgrade an Existing SP3 or SP4 Domain or Create A New SP5 Domain

You must choose one of the following upgrade paths:

- Upgrade your existing SP3 or SP4 domain to use the new SP5 libraries and database modules.

  Typically you would use this upgrade path if you want to preserve a highly customized domain, including LDAP setup.

- Create a new domain with the SP5 Configuration Wizard that mirrors your existing SP3 or SP4 domain.

  Typically you would use this path if you are comfortable recreating your domain configuration, and you want to ensure that your domain is based on the latest domain template. If you are using this method, skip to "Creating a New SP5 Domain" on page 7-5.

Upgrading an Existing SP3 or SP4 Domain

If you choose to upgrade your existing SP3 or SP4 domain to use the new SP5 libraries, upgrade instructions vary according to the method you used to install SP5. The following sections explain your options and the required manual update steps.

WebLogic Portal Upgrade Installer

If you used the WebLogic Portal SP5 upgrade installer to update your existing SP3 or SP4 install, then you do not need to make any changes to your SP3 or SP4 domain. All scripts should pick up the appropriate new libraries for the SP5 environment.

Continue to "Step 2: Upgrade Existing Portal Framework Data" on page 7-5.

Update Paths to JDK as Needed

The upgrade installer cannot update paths to the JDK because it can reside in a unique location. To update paths to point to the new JDK for SP5, edit the variables in the scripts shown in Table 7 -7:

Table 7 -7 Variables/Scripts Containing Path

Variable	Script
JDK_HOME BEAHOME WEBLOGIC_HOME PORTAL_HOME POINTBASE_HOME	set-dbenv
JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JDK_HOME	setDomainEnv

Edit URL Template File for Use with WSRP

The `url-template-config.xml` file, which is automatically created in a portal Web project, contains URL templates and variables for WSRP portlets. You must make the following two changes in this file for each WSRP URL template, so that the BEA-provided consumer will be able to honor mode changes and state changes from non-BEA producers:

- Replace `_mode` with `wsrp-mode`
- Replace `_state` with `wsrp-windowState`

In addition, the producer should use the `wsrp:` prefix for standard modes and window states in URLs, so that non-BEA consumers can recognize URLs returned by BEA-provided producers.

WebLogic Portal Full Installer

If you used the full installer and you installed SP5 in a separate directory, then you must manually update scripts in your SP3 or SP4 domain.

Update setDomainEnv

In the script `setDomainEnv`, update the following variables as needed:

```
JAVA_HOME
BEA_JAVA_HOME
SUN_JAVA_HOME
WL_HOME
```

Note: The file `setDomainEnv` contains the `DOMAIN_NAME` variable. Do not change the value of this variable if you are upgrading an existing domain to SP5.

Update Path to JDK as Needed

You might need to update the path to point to the new JDK for SP5. The variables and their corresponding scripts are shown in Table 7 -8:

Table 7 -8 Variables and Scripts Containing Path

Variable	Script/File
JDK_HOME JDK_TOOLS	set-dbenv
JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JAVA_VENDOR JDK_HOME	setDomainEnv

Review and Update Other Scripts and Variables as Needed

The following table lists other variables and files that may or may not need updating, depending on which scripts you use in your environment; however, BEA recommends that you review the files in your existing domain directory to verify that you have made all needed changes.

Table 7 -9 Other Variables and Files

Variable	Script/File	Notes
BEAHOME WL_HOME JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JDK_HOME POINTBASE_HOME	webappCompile startPointBaseConsole startWebLogic[1] stopManagedWebLogic stopWebLogic	[1]The file `startWebLogic` contains the `DOMAIN_NAME` variable. Do not change the value of this variable if you are upgrading an existing domain to SP5.
wlsHome.path jdkHome.path	workshop.properties	

Creating a New SP5 Domain

To create a new SP5 domain that mirrors your existing SP3 or SP4 domain, follow these steps:

1. Use the Configuration Wizard.

   See "Creating a New WebLogic Domain" at:
   http://edocs.bea.com/platform/docs81/confgwiz/newdom.html.

2. Copy the LDAP information from your SP3 or SP4 domain to the new SP5 domain. LDAP information is located in the path: *portal_domain*\portalServer\ldap.

Step 2: Upgrade Existing Portal Framework Data

Warning: If you are upgrading from SP3 to SP5, you must first follow the steps to upgrade your database from SP3 to SP4 as described in Chapter 6, "Upgrading to Service Pack 4." After following those steps, you can then follow the steps in this chapter to upgrade your database data to SP5.

The script that you run as part of this step updates the default markup in the database. The default markup for books, pages, and desktops is used when creating them with the WebLogic Portal Administration Portal. The script includes the token `$(markupname)` in the database. If this token is not added, SAXParseException will be thrown when you import desktops, pages, or books that were created with the Administration Portal or using Visitor Tools.

Scripts for upgrading SP4 database data to SP5 are located in *SP5_WL_HOME*\portal\upgrade\SP5 *where* *SP5_WL_HOME* is the *WL_HOME* directory for SP5.

For databases other than PointBase, in order to use the provided WebLogic Portal upgrade scripts, database client software must be installed and configured on the machine from which you execute the upgrade scripts.

Complete the following steps to make your WebLogic Platform 8.1 SP4 database data SP5-compliant. These steps are identical for all database types.

1. Shut down the WebLogic Server if it is running. The WebLogic Server must be shut down while performing the database data upgrade steps.

2. Back up your existing database using a method prescribed by your database vendor. Ensure that the database backup you create is valid for restore purposes if you encounter a failure during the database upgrade process.

3. Navigate to *SP5_WL_HOME*\portal\upgrade\SP5 and run the `pf_update_system_data.sql` script to upgrade your database data.

The database is now upgraded. Back up the upgraded database before upgrading existing applications.

Step 3: Upgrade Existing Applications

When you upgrade from SP4 to SP5, you must change the following configurations for your application:

- Re-configure your third-party content management system, if applicable.
- Re-set the passwords for your WSRP portlets, if applicable
- Update the portal libraries in any applications you developed. Updating overwrites the existing libraries.

Re-Configure Third-Party Content Management Systems

If you are using a third-party content management system, you'll need to re-configure that repository before deploying. This allows the passwords to be re-encrypted for the new domain. If you are not switching domains when upgrading, you may skip this task.

First, remove the existing repository connection, using the Portal Administration Tools:

1. View the **Repository List** tree by selecting Repository from the View drop-down list.
2. In the **Manage Repositories** tree, right-click the repository you want to remove.
3. Select **Remove** from the pop-up menu.

   Note: Be sure to write down all the configuration information you will need to re-enter.
4. A pop-up window displays confirming that you want to remove the repository. If you want to proceed, click `OK`.

Then, re-connect to the repository.

1. View the **Repository List** tree by selecting Repository from the View drop-down list.
2. In the **Manage Repositories** tree, right click the Virtual Content Repository.
3. Select `Add Repository` from the pop-up menu.
4. In the `Editor` pane, provide the connection information
5. In the **Cache Properties** box, edit the cache properties, if necessary. You can also choose to accept the default cache settings.

6. Click **Create**.

Re-Encrypting Passwords for WSRP Producers

If you are moving to a new domain when upgrading, you'll need to re-encrypt any passwords that are used to access WSRP portlets. Encryption is domain-specific so it needs to be updated when switching domains.

If you are deploying your application as an EAR file, this needs to be done manually with the `EncryptDomainString` command-line utility which is used to generate an encrypted password and then place that encrypted password into the `application-config.xml` file before you deploy your application. The application must already exist in the new domain.

Note: If you are going to deploy an exploded application (not an EAR), you can use the Service Administration tool in Portal Administration Portal to re-set passwords. When passwords are re-entered using the Service Administration tool, they are automatically encrypted. However, the Portal Administration Tool cannot be used if you have already compressed your application to an EAR file.

A portal application's `META-INF` directory contains the respective `application-config.xml`. file where WSRP passwords are stored automatically. For example, Listing 7-1 shows the WSRP element from `application-config.xml` during the development phase (exploded) before deployment as an EAR file:

Listing 7-1 Development Phase Clear Text Passwords in `application-config.xml`

```
<ConsumerSecurity AdminPassword="weblogic" AdminUserName="weblogic"
   CertAlias="wsrpConsumer" CertPrivateKeyPassword="wsrppassword"
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword="password"
   Name="ConsumerSecurity"/>
```

After the Service Administration tool is used to edit attributes, the file is saved and passwords are automatically encrypted, as shown in Listing 7-2:

Listing 7-2 Encrypted Passwords in `application-config.xml`

```
<ConsumerSecurity AdminPassword="{3DES}3QrrUeIwN/DxlDI++1ixPw=="
   AdminUserName="weblogicc" CertAlias="wsrpConsumer"
   CertPrivateKeyPassword="{3DES}g7h+VOSAsO9pSlvYSSB2iw=="
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword=
      "{3DES}1OLYVirMWOo+3sEU80cMqw=="

   Name="ConsumerSecurity" />
```

Encrypting Passwords

When upgrading applications that are stored as EAR files, you must manually update the encryption.

To encrypt passwords, use this procedure:

1. Open a command box (DOS shell) and navigate to *domain*/portal/ (where *domain* is the domain directory for the application) and run `setDomainEnv.cmd`.
2. At the prompt, enter

   ```
   java com.bea.p13n.util.EncryptDomainString -targetDomainDir d
   -inputString s
   ```

   where:

   – *d* is the domain directory to which the portal application is being deployed; for example,

   – *s* is the input password to encrypt

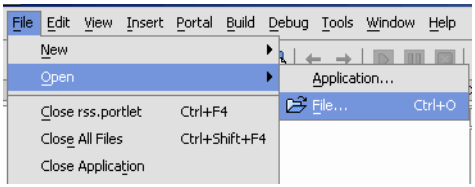
   For example:

   ```
   java com.bea.p13n.util.EncryptDomainString -targetDomainDir
   \bea\weblogic81b\samples\domain\portal -inputString weblogic
   ```

   In this example, the input string `weblogic` represents administrator's password (`adminpassword=weblogic`; see Listing 7-3). The command line utility prints a domain specific encrypted string.
3. Open WebLogic Workshop and the specific portal application.

4. Select File>Open>File (Figure 7-1).

Figure 7-1 Opening a File in WebLogic Workshop



An Open dialog box appears.

5. Navigate to the application's `META-INF` folder and open `application-config.xml`.

Listing 7-3 Clear text Passwords in application-config.xml

```
<ConsumerSecurity AdminPassword="weblogic" AdminUserName="weblogic"
   CertAlias="wsrpConsumer" CertPrivateKeyPassword="wsrppassword"
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword="password"
   Name="ConsumerSecurity"/>
```

6. Replace the previously encrypted passwords with those generated by the `EncryptDomainString` utility, as shown in Figure 7-4. You will need to run `EncryptDomainString` for each password in the `<ConsumerSecurity>` (Listing 7-3) element; for example:
   - `AdminPassword="weblogic"`
   - `CertPrivateKeyPassword="wsrppassword"`
   - `KeystorePassword="password"`

Listing 7-4 Encrypted Passwords Generated by EncryptDomainString Utility

```
<ConsumerSecurity AdminPassword="{3DES}3QrrUeIwN/DxlDI++1ixPw=="
   AdminUserName="weblogicc" CertAlias="wsrpConsumer"
   CertPrivateKeyPassword="{3DES}g7h+VOSAsO9pSlvYSSB2iw=="
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
```

```
   DefaultIdentityAssertionProvider"
Keystore="wsrpKeystore.jks" KeystorePassword=
   "{3DES}1OLYVirMWOo+3sEU80cMqw=="

Name="ConsumerSecurity" />
```

Note on Changing Passwords

If you need to change an administrator's password for any reason, simply changing the password will result in having to rebuild and redeploy the EAR. This is time-consuming and counterproductive. Instead, you can work around this problem by doing the following:

1. Create a special user in the target system for a WSRP administrator. See Create a New User for more information on creating a user.
2. Make that user a member of the administrator group. See Add a User to a Group for more information on adding a member to a group.
3. Insert the new logon information (username and password) into the application's `application-config.xml` file as described in Encrypting Passwords.

Updating Portal Libraries

After you install a new service pack that includes portal library updates, you must update the libraries in the applications you have developed. Updating overwrites the existing libraries.

Before You Begin – About UUP

If you have developed your own Unified User Profile (UUP) to access user profile properties stored in an external user store, you have most likely modified and re-created the `p13n_ejb.jar` file in your application root directory. Because `p13n_ejb.jar` is one of the files overwritten in the following procedure, you should back up your existing file. After the upgrade procedure, you must re-create the updated `p13n_ejb.jar` with your UUP implementation. For more information, see "Setting up Unified User Profiles" in the *User Management Guide* at http://e-docs.bea.com/wlp/docs81/users/uup.html#999527.

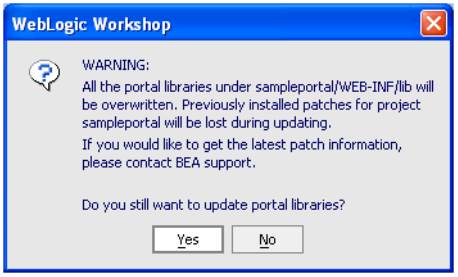
Update the Portal Libraries

To update your application libraries, follow these steps:

1. Locate and delete the `.workshop` directory from the application that you will be working with. Do this prior to starting WebLogic Workshop.
2. Start WebLogic Workshop 8.1 SP5.
3. If your WebLogic server is running, shut it down by choosing, in WebLogic Workshop, **Tools→WebLogic Server→Stop WebLogic Server**.
4. In WebLogic Workshop SP5, open the portal application that you want to update; for example, navigate to the SP4 or SP3 Portal application.
5. In the Application window, right-click the application directory and choose **Install→Update Portal Libraries**.

   After the portal application libraries are updated, a window appears; continue to the next step.
6. In the displayed window, select the Web projects whose libraries you want to update, and click **OK**.

   For each selected Web project, the following warning window appears:



   Click **Yes** to continue with the update.

   If you choose not to update a Web project's libraries using the displayed window, you can update the Web project later by right-clicking the Web project directory in the Application window and choosing **Install→Update Portal Libraries**.
7. If your application uses Commerce or Pipeline components, right-click the application directory and choose **Install→Commerce Services** and **Install→Pipeline Services**.
8. If your application uses Commerce or Webflow JSP tag libraries, right-click the Web project directory in the Application window and choose **Install→Commerce Taglibs** and **Install→Webflow Taglibs**.

9. If you have hidden any Web applications in the WebLogic Workshop interface, those Web applications will not be updated. Either un-hide them and perform the update as described in the previous steps, or manually replace the updated libraries in the hidden Web application(s).

   To apply the service pack library updates to portal applications already deployed in production, redeploy those applications after you have updated them in WebLogic Workshop. See "Step 4: Redeploy the Upgraded Application" for deployment instructions.

10. Update your Portal server settings if needed by selecting Tools→WebLogic Server→Server Properties and editing this information.

11. If you developed your own Unified User Profile (UUP) by modifying `p13n_ejb.jar` in your application root directory, re-implement your UUP in the new `p13n_ejb.jar` file. See "Before You Begin – About UUP" on page 7-10.

12. Rebuild the application by selecting Build→Build Application.

13. Restart the server in the upgraded domain.

Step 4: Redeploy the Upgraded Application

The final upgrade step is to redeploy the application on your server.

For deployment instructions, see the *WebLogic Portal Production Operations Guide* at

http://edocs.bea.com/wlp/docs81/prodOps/index.html.

Step 5: Review Functional Changes for SP5

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 8

Upgrading to Service Pack 6

This section provides instructions for applying Service Pack 6 (SP6) changes to your portal applications after you install that service pack. For instructions on installing service packs, see "Installing Service Packs and Rolling Patches" at:

`http://e-docs.bea.com/platform/docs81/install/update.html`

This section contains information on the following subjects:

- Step 1: Upgrade an Existing SP4 or SP5 Domain or Create A New SP6 Domain
- Step 2: Upgrade Existing Applications
- Step 3: Redeploy the Upgraded Application
- Step 4: Review Functional Changes for SP6

For detailed information on functional changes that might require you to perform manual tasks or configuration changes, see the following sections:

- Functional Changes Affecting Your WebLogic Portal Environment

The Service Pack Upgrade Process

Regardless of the release—8.1 or 8.1 with a previous service pack—from which you are upgrading, you follow the same basic process to complete the upgrade.

1. Upgrade an existing SP4 or SP5 domain, or create a new SP6 domain.
2. Upgrade existing database data.

Note: No database schema changes are required for Service Pack 6; however, you must update the default markup in the database, using the provided script, to be SP6 compliant.

3. Upgrade existing applications.
4. Redeploy upgraded application to your production environment.
5. Perform manual upgrade tasks to enable SP6 enhancements, if applicable.

The following sections describe these steps.

Step 1: Upgrade an Existing SP4 or SP5 Domain or Create A New SP6 Domain

You must choose one of the following upgrade paths:

- Upgrade your existing SP4 or SP5 domain to use the new SP6 libraries and database modules.

  Typically you would use this upgrade path if you want to preserve a highly customized domain, including LDAP setup.

- Create a new domain with the SP6 Configuration Wizard that mirrors your existing SP4 or SP5 domain.

  Typically you would use this path if you are comfortable recreating your domain configuration, and you want to ensure that your domain is based on the latest domain template. If you are using this method, skip to "Creating a New SP6 Domain" on page 8-5.

Upgrading an Existing SP4 or SP5 Domain

If you choose to upgrade your existing SP4 or SP5 domain to use the new SP6 libraries, upgrade instructions vary according to the method you used to install SP6. The following sections explain your options and the required manual update steps.

WebLogic Portal Upgrade Installer

If you used the WebLogic Portal SP6 upgrade installer to update your existing SP4 or SP5 install, then you do not need to make any changes to your SP4 or SP5 domain. All scripts should pick up the appropriate new libraries for the SP6 environment.

Continue to "Step 2: Upgrade Existing Applications" on page 8-5.

Update Paths to JDK as Needed

The upgrade installer cannot update paths to the JDK because it can reside in a unique location. To update paths to point to the new JDK for SP5, edit the variables in the scripts shown in Table 8 -10:

Table 8 -10 Variables/Scripts Containing Path

Variable	Script
JDK_HOME BEAHOME WEBLOGIC_HOME PORTAL_HOME POINTBASE_HOME	set-dbenv
JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JDK_HOME	setDomainEnv

Edit URL Template File for Use with WSRP

The `url-template-config.xml` file, which is automatically created in a portal Web project, contains URL templates and variables for WSRP portlets. You must make the following two changes in this file for each WSRP URL template, so that the BEA-provided consumer will be able to honor mode changes and state changes from non-BEA producers:

- Replace `_mode` with `wsrp-mode`
- Replace `_state` with `wsrp-windowState`

In addition, the producer should use the `wsrp:` prefix for standard modes and window states in URLs, so that non-BEA consumers can recognize URLs returned by BEA-provided producers.

WebLogic Portal Full Installer

If you used the full installer and you installed SP6 in a separate directory, then you must manually update scripts in your SP4 or SP5 domain.

Update setDomainEnv

In the script `setDomainEnv`, update the following variables as needed:

```
JAVA_HOME
BEA_JAVA_HOME
```

SUN_JAVA_HOME
WL_HOME

Note: The file setDomainEnv contains the DOMAIN_NAME variable. Do not change the value of this variable if you are upgrading an existing domain to SP6.

Update Path to JDK as Needed

You might need to update the path to point to the new JDK for SP6. The variables and their corresponding scripts are shown in Table 8 -11:

Table 8 -11 Variables/Scripts Containing Path

Variable	Script/File
JDK_HOME JDK_TOOLS	set-dbenv
JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JAVA_VENDOR JDK_HOME	setDomainEnv

Review and Update Other Scripts and Variables as Needed

The following table lists other variables and files that may or may not need updating, depending on which scripts you use in your environment; however, BEA recommends that you review the files in your existing domain directory to verify that you have made all needed changes.

Table 8 -12 Other Variables and Files

Variable	Script/File	Notes
BEAHOME WL_HOME JAVA_HOME BEA_JAVA_HOME SUN_JAVA_HOME JDK_HOME POINTBASE_HOME	webappCompile startPointBaseConsole startWebLogic[1] stopManagedWebLogic stopWebLogic	[1]The file startWebLogic contains the DOMAIN_NAME variable. Do not change the value of this variable if you are upgrading an existing domain to SP6.
wlsHome.path jdkHome.path	workshop.properties	

Creating a New SP6 Domain

To create a new SP6 domain that mirrors your existing SP4 or SP5 domain, follow these steps:

1. Use the Configuration Wizard.

   See "Creating a New WebLogic Domain" at:
   http://edocs.bea.com/platform/docs81/confgwiz/newdom.html.

2. Copy the LDAP information from your SP4 or SP5 domain to the new SP6 domain. LDAP information is located in the path: *portal_domain*\portalServer\ldap.

Step 2: Upgrade Existing Applications

When you upgrade from SP5 to SP6, you must change the following configurations for your application:

- Re-configure your third-party content management system, if applicable.
- Re-set the passwords for your WSRP portlets, if applicable
- Update the portal libraries in any applications you developed. Updating overwrites the existing libraries.

Re-Configure Third-Party Content Management Systems

If you are using a third-party content management system, you'll need to re-configure that repository before deploying. This allows the passwords to be re-encrypted for the new domain. If you are not switching domains when upgrading, you may skip this task.

First, remove the existing repository connection, using the Portal Administration Tools:

1. View the **Repository List** tree by selecting Repository from the View drop-down list.
2. In the **Manage Repositories** tree, right-click the repository you want to remove.
3. Select **Remove** from the pop-up menu.

   Note: Be sure to write down all the configuration information you will need to re-enter.

4. A pop-up window displays confirming that you want to remove the repository. If you want to proceed, click OK.

Then, re-connect to the repository.

1. View the **Repository List** tree by selecting Repository from the View drop-down list.

2. In the **Manage Repositories** tree, right click the Virtual Content Repository.
3. Select `Add Repository` from the pop-up menu.
4. In the `Editor` pane, provide the connection information
5. In the **Cache Properties** box, edit the cache properties, if necessary. You can also choose to accept the default cache settings.
6. Click **Create**.

Re-Encrypting Passwords for WSRP Producers

If you are moving to a new domain when upgrading, you'll need to re-encrypt any passwords that are used to access WSRP portlets. Encryption is domain-specific so it needs to be updated when switching domains.

If you are deploying your application as an EAR file, this needs to be done manually with the `EncryptDomainString` command-line utility which is used to generate an encrypted password and then place that encrypted password into the `application-config.xml` file before you deploy your application. The application must already exist in the new domain.

Note: If you are going to deploy an exploded application (not an EAR), you can use the Service Administration tool in Portal Administration Portal to re-set passwords. When passwords are re-entered using the Service Administration tool, they are automatically encrypted. However, the Portal Administration Tool cannot be used if you have already compressed your application to an EAR file.

A portal application's `META-INF` directory contains the respective `application-config.xml`. file where WSRP passwords are stored automatically. For example, Listing 8-1 shows the WSRP element from `application-config.xml` during the development phase (exploded) before deployment as an EAR file:

Listing 8-1 Development Phase Clear Text Passwords in `application-config.xml`

```
<ConsumerSecurity AdminPassword="weblogic" AdminUserName="weblogic"
   CertAlias="wsrpConsumer" CertPrivateKeyPassword="wsrppassword"
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword="password"
   Name="ConsumerSecurity"/>
```

After the Service Administration tool is used to edit attributes, the file is saved and passwords are automatically encrypted, as shown in Listing 8-2:

Listing 8-2 .Encrypted Passwords in `application-config.xml`

```
<ConsumerSecurity AdminPassword="{3DES}3QrrUeIwN/DxlDI++1ixPw=="
   AdminUserName="weblogicc" CertAlias="wsrpConsumer"
   CertPrivateKeyPassword="{3DES}g7h+VOSAsO9pSlvYSSB2iw=="
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword=
      "{3DES}1OLYVirMWOo+3sEU80cMqw=="

   Name="ConsumerSecurity" />
```

Encrypting Passwords

When upgrading applications that are stored as EAR files, you must manually update the encryption.

To encrypt passwords, use this procedure:

1. Open a command box (DOS shell) and navigate to *domain*`/portal/` (where *domain* is the domain directory for the application) and run `setDomainEnv.cmd`.
2. At the prompt, enter

   ```
   java com.bea.p13n.util.EncryptDomainString -targetDomainDir d
   -inputString s
   ```

   where:

   – *d* is the domain directory to which the portal application is being deployed; for example,

   – *s* is the input password to encrypt

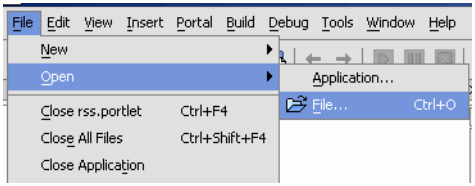
   For example:

   ```
   java com.bea.p13n.util.EncryptDomainString -targetDomainDir
   \bea\weblogic81b\samples\domain\portal -inputString weblogic
   ```

In this example, the input string `weblogic` represents administrator's password (`adminpassword=weblogic`; see Listing 8-3). The command line utility prints a domain specific encrypted string.

3. Open WebLogic Workshop and the specific portal application.
4. Select **File > Open > File** (Figure 8-1).

Figure 8-1 Opening a File in WebLogic Workshop



An Open dialog box appears.

5. Navigate to the application's `META-INF` folder and open `application-config.xml` file.

Listing 8-3 Clear Text Passwords in application-config.xml

```
<ConsumerSecurity AdminPassword="weblogic" AdminUserName="weblogic"
   CertAlias="wsrpConsumer" CertPrivateKeyPassword="wsrppassword"
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword="password"
   Name="ConsumerSecurity"/>
```

6. Replace the previously encrypted passwords with those generated by the `EncryptDomainString` utility, as shown in Figure 8-4. You will need to run `EncryptDomainString` for each password in the `<ConsumerSecurity>` (Listing 8-3) element; for example:
   - `AdminPassword="weblogic"`
   - `CertPrivateKeyPassword="wsrppassword"`
   - `KeystorePassword="password"`

Listing 8-4 Encrypted Passwords Generated by EncryptDomainString Utility

```
<ConsumerSecurity AdminPassword="{3DES}3QrrUeIwN/DxlDI++1ixPw=="
   AdminUserName="weblogicc" CertAlias="wsrpConsumer"
   CertPrivateKeyPassword="{3DES}g7h+VOSAsO9pSlvYSSB2iw=="
   ConsumerName="wsrpConsumer"
   IdentityAssertionProviderClass="com.bea.wsrp.security.
      DefaultIdentityAssertionProvider"
   Keystore="wsrpKeystore.jks" KeystorePassword=
      "{3DES}1OLYVirMWOo+3sEU80cMqw=="

   Name="ConsumerSecurity" />
```

Note on Changing Passwords

If you need to change an administrator's password for any reason, simply changing the password will result in having to rebuild and redeploy the EAR. This is time-consuming and counterproductive. Instead, you can work around this problem by doing the following:

1. Create a special user in the target system for a WSRP administrator. See Create a New User for more information on creating a user.
2. Make that user a member of the administrator group. See Add a User to a Group for more information on adding a member to a group.
3. Insert the new logon information (username and password) into the application's `application-config.xml` file as described in Encrypting Passwords.

Updating Portal Libraries

After you install a new service pack that includes portal library updates, you must update the libraries in the applications you have developed. Updating overwrites the existing libraries.

Before You Begin – About UUP

If you have developed your own Unified User Profile (UUP) to access user profile properties stored in an external user store, you have most likely modified and re-created the `p13n_ejb.jar` file in your application root directory. Because `p13n_ejb.jar` is one of the files overwritten in the following procedure, you should back up your existing file. After the upgrade procedure, you must re-create the updated `p13n_ejb.jar` with your UUP implementation. For more

information, see “Setting up Unified User Profiles” in the *User Management Guide* at http://e-docs.bea.com/wlp/docs81/users/uup.html#999527.

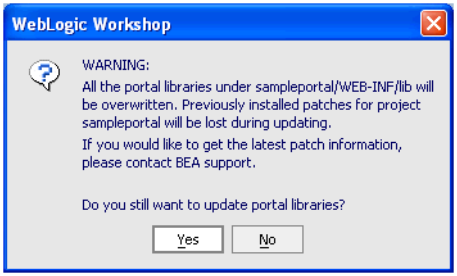
Update the Portal Libraries

To update your application libraries, follow these steps:

1. Locate and delete the `.workshop` directory from the application that you will be working with. Do this prior to starting WebLogic Workshop.
2. Start WebLogic Workshop 8.1 SP6.
3. If your WebLogic server is running, shut it down by choosing, in WebLogic Workshop, **Tools > WebLogic Server > Stop WebLogic Server**.
4. In WebLogic Workshop SP6, open the portal application that you want to update; for example, navigate to the SP5 or SP4 Portal application.
5. In the Application window, right-click the application directory and choose **Install > Update Portal Libraries**.

   After the portal application libraries are updated, a window appears; continue to the next step.
6. In the displayed window, select the Web projects whose libraries you want to update, and click **OK**.

   For each selected Web project, the following warning window appears:



   Click **Yes** to continue with the update.

   If you choose not to update a Web project's libraries using the displayed window, you can update the Web project later by right-clicking the Web project directory in the Application window and choosing **Install > Update Portal Libraries**.

7. If your application uses Commerce or Pipeline components, right-click the application directory and choose **Install > Commerce Services** and **Install > Pipeline Services**.
8. If your application uses Commerce or Webflow JSP tag libraries, right-click the Web project directory in the Application window and choose **Install > Commerce Taglibs** and **Install > Webflow Taglibs**.
9. If you have hidden any Web applications in the WebLogic Workshop interface, those Web applications will not be updated. Either un-hide them and perform the update as described in the previous steps, or manually replace the updated libraries in the hidden Web applications.

   To apply the service pack library updates to portal applications already deployed in production, redeploy those applications after you have updated them in WebLogic Workshop. See "Step 3: Redeploy the Upgraded Application" for deployment instructions.
10. Update your Portal server settings if needed by selecting Tools > WebLogic Server > Server Properties and editing this information.
11. If you developed your own Unified User Profile (UUP) by modifying `p13n_ejb.jar` in your application root directory, re-implement your UUP in the new `p13n_ejb.jar` file. See "Before You Begin – About UUP" on page 8-9.
12. Rebuild the application by selecting Build > Build Application.
13. Restart the server in the upgraded domain.

Step 3: Redeploy the Upgraded Application

The final upgrade step is to redeploy the application on your server.

For deployment instructions, see the *WebLogic Portal Production Operations Guide* at

http://edocs.bea.com/wlp/docs81/prodOps/index.html.

Step 4: Review Functional Changes for SP6

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.

APPENDIX A

Functional Changes Affecting Your WebLogic Portal Environment

This appendix describes functional changes in WebLogic Portal Version 8.1 through 8.1 SP5 that affect your upgraded environment and might require that you perform manual tasks.

- Portlet State Persistence (Base Release, SP5)
- New Option for Request Attribute Persistence (SP4)
- New Generic URL Tag (SP6)
- Oracle 10g R2 Database Requires Manual Privilege Setup
- Associating Portlets with Pages (Base Release, SP5)
- Mandatory Portlets (8.1 Base Release)
- Upgrading Simple Producers from Service Pack 3
- Aggressive Control Tree Persistence (SP4)
- Content Management Portlet (SP4)
- URL Template File Updates (SP4)
- Portlet Preferences Behavior on Server Restart (SP4)
- Entitlements Data Source (SP4)
- Storage of Policy Reference Data in RDBMS (SP4)
- Using Role Caching with Entitlements (SP4)

- Setting up JDBC Connection Polling (SP4)
- Portal Projects in Source Control (SP4)
- Using New Proxy Portlet Cache (SP6)
- Using Content Query Expressions (SP6)
- Propagating Request Parameters from WSRP Requests (SP6)

For additional information on new features in WebLogic Portal, see the WebLogic Platform Release Notes at:

http://edocs.bea.com/platform/docs81/relnotes/relnotes.html.

Portlet State Persistence (Base Release, SP5)

In WebLogic Portal 7.0, minimized portlet states were persisted in the database, but in WebLogic Portal 8.1 and service packs they are persisted only for the session. This section describes a way to allow database persistence of the portlet state, based on the following new API method that was added in the WindowBackingContext in Service Pack 5:

```
public void setupStateChangeEventFromParent(String stateValue)
```

You can use this method in a backing file to change the state of portlets from a parent control, such as a desktop, book, or page. This way, one backing file at the desktop level can control the states of all portlets under the desktop.

A sample backing file that performs this task is provided as a sample in the dev2dev area of the BEA web site. The implementation described in this section is based on the sample backing file. Because the code for this solution is provided in a backing file, this code can be changed/updated as needed within your application.

This implementation utilizes the EntityPropertyManager to store the portlet states in the PROPERTY_KEY and PROPERTY_VALUE tables. The property names are set dynamically using the portal, desktop, and portlet labels and are mapped to each user who changes the state.

To set up portlet state persistence using the sample backing file, follow these steps:

1. In WebLogic Workshop, create a property set named `portletStateValue`.
   a. Right-click `portalApp/data/userprofiles` and select **New > User Profile Property Set**.
   b. Name the property set `portletStateValue.usr` and click **Create**.

Nothing else is required for this property set—you will use it as a key for storing the portlet state.

This `portletStateValue` key is used in the backing file as a static final string variable. You can change the name of the property set if desired; if you do so, remember to update the string in the backing file.

2. Create the following directory under your portal web application:

   `web-inf/src/backing/statemgmt`

3. Copy the dev2dev sample Java backing file (PortletStateMgmtDesktopBacking.java) to the directory you created in the previous step.

   This file contains comments explaining the logic for persisting and restoring the portlet state.

4. In WebLogic Workshop, open the .portal file and select the desktop. In the property editor, enter the following for **Backing File**:

   `backing.statemgmt.PortletStateMgmtDesktopBacking`

Important note: This implementation was tested using the sample.portal that is provided by default, and uses the login portlet example. When logging in, the handlePostbackData method is called in the backing file, and `_nfpb=true` is set in the URL. If you are using a different login mechanism, make sure handlePostbackData is called when the portal is being rendered the first time after login.

New Option for Request Attribute Persistence (SP4)

In Version 8.1 SP4, a request persistence optimization enhancement led to the addition of a new attribute for page flow and struts portlets. The new `requestAttrPersistence` attribute allows you to capture the state of the request immediately after an action has run, so that a JSP can depend on that state, and can be rendered again any number of times to support page refreshes.

Although the default value of `session` in the new attribute preserves the existing behavior, if you attempt to modify the state of the request within the JSP, the new behavior will not allow this implementation to work. Ideally, you should move any logic that modifies the state of the request out of the JSP and into the action. A less desirable workaround would be to set the Request Attribute Persistence attribute to `none` to disable WebLogic Portal's automatic persistence functionality.

For more information about the Request Attribute Persistence attribute, refer to the WebLogic Workshop online help at:

http://e-docs.bea.com/workshop/docs81/doc/en/portal/buildportals/portalPropertiesPortlet.html

New Generic URL Tag (SP6)

In WebLogic Portal SP6, a new method is available for the GenericURL class to override the `<generate-xml-amp-entity>` attribute in the `url-template-config.xml` file. This method was created to allow for programmatic use of URLs.

The new `GenericURL.setForcedAmpForm(boolean)` provides a public API to set the URL query separator. This overrides the default mechanisms for determining ampersand entity or character usage. Passing in `true` generates the URL with ampersand entities ("&") and passing in `false` generates the URL with the ampersand character ("&").

The following code fragment shows how to override a true value in the `<generate-xml-amp-entity>` element:

```
PageURL postUrl = PageURL.createPageURL(getRequest(), getResponse());
   postUrl.setPageLabel("Test_page_6");
   postUrl.setForcedAmpForm(false);
   postUrl.addParameter("test","test",true);
```

The resulting URL would look like this:

```
http://localhost:7001/test/test.portal?_nfpb=true&;_pageLabel=Test_page_6&
test=test
```

A corresponding attribute for the GenericURL JSP tag, called `forcedAmpForm`, was also added. This new attribute was added to the following tags:

- `render:windowUrl`
- `render:standalonePortletUrl`
- `render:pageUrl`
- `render:postbackUrl`
- `render:resourceUrl`
- `render:toggleButtonUrl`
- `render:jspContentUrl`

The following code fragment is from the tag library definition for each of the above JSP tags:

```
<attribute
   <name>forcedAmpForm</name>
   <required>false</required>
   <rtexprvalue>true</rtexprvalue>
```

```
    <description>Forces the form of query separators used for this URL.
    This overrides the natural mechanisms for determining ampersand entity
    or character usage based on document type and/or configuration.
    If the value 'true' is used, then "&amp;" will be the separator.
    If 'false' is used, then the "&" character will be the
    separator.</description>
</attribute>
```

Oracle 10g R2 Database Requires Manual Privilege Setup

As described in Oracle Note 317258.1, Oracle changed their use of predefined database roles in the 10g R2 version to provide more security. The CONNECT role now provides only CREATE SESSION privileges; the other associated privileges have been removed.

You must now explicitly grant the CREATE VIEW privilege for Oracle 10g R2 databases.

Associating Portlets with Pages (Base Release, SP5)

In WebLogic Portal 7.0, the portal administrator had the ability to restrict portlets so that they could display only on particular pages. In WebLogic Portal 8.1, this functionality is not directly available because users can now utilize visitor tools to display portlets (to which they are entitled) on any available page. This section describes a way to allow portlet association with particular pages, based on the following new method that was added to PortalVisitorManager in Service Pack 5:

```
public static String[] generateOnePortletCategoryArray(String webAppName,
String pageLabel, String indentStr, HttpServletRequest request)
```

The new method in Service Pack 5 that achieves similar functionality to that of WebLogic Portal 7.0 uses portlet categories and modified visitor tools. The administrator places portlets into a category and associates it using a unique page label; the visitor tools then filter the portlet list to include only those in the appropriate category.

Note: To use this implementation, all pages within a portal with visitor tools customization/capability must be assigned a portlet category to allow the code to find the portlets for the individual pages.

You can view a code example that uses portlet categories and modified visitor tools on the dev2dev web site at: https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=S174.

Mandatory Portlets (8.1 Base Release)

In WebLogic Portal 7.0, the portal administrator had the ability set a portlet as “Mandatory” to prevent certain portlets from being removed as a result of a user's customizations; however, using the “Mandatory” flag in 7.0 did not prevent a user from (for example) moving the portlet to the bottom of a page, which could result in the portlet not being visible without scrolling down the page.

In WebLogic Portal 8.1, locking placeholders were introduced to require a fixed position of the portlet on a page. (For instructions on using locking placeholders, see the Administration Console online help at http://e-docs.bea.com/wlp/docs81/adminportal/help/PM_PortletLock.html.)

To create *mandatory* portlets that appear in a *specific position* on a web page, you can use a combination of a locking placeholder and a custom layout; you lock a placeholder for a column in the layout for the page, and place mandatory portlets in that placeholder. You use custom layouts to create the layout you need for this purpose.

The following white paper includes instructions on creating custom layouts:

http://e-docs.bea.com/wlp/docs81/whitepapers/netix/body.html

Upgrading Simple Producers from Service Pack 3

When upgrading a simple producer from WebLogic Portal Service Pack 3 to a later service pack, WSRP producer web application libraries are not automatically updated. If you upgrade from SP3 to a later service pack, you must perform the following steps to ensure that your simple producer continues to function properly as a WSRP producer.

Note: A simple producer is a non-portal web application. For more information on simple producers, see “Producers” in *Using WSRP with* WebLogic Portal.

1. Perform the normal procedures for upgrading a WebLogic Portal application from SP3 to a later service pack.
2. Copy the following files into the `WEB-INF/lib` directory of your web application:

```
BEAHOME/weblogic81/portal/lib/netuix/web/netui-adapter.jar
BEAHOME/weblogic81/portal/lib/netuix/system/ext/web/struts-adapter.jar
BEAHOME/weblogic81/portal/lib/netuix/system/ext/web/struts-adapter-html.tld
BEAHOME/weblogic81/portal/lib/netuix/system/ext/web/struts-adapter-naming.tld
BEAHOME/weblogic81/portal/lib/netuix/system/ext/web/struts-adapter-nested.tld
BEAHOME/weblogic81/portal/lib/netuix/system/ext/web/struts-adapter-tiles.tld
BEAHOME/weblogic81/portal/lib/wsrp/wsrp-producer.jar
BEAHOME/weblogic81/portal/lib/wsrp/adapters/wsrp-jpf-adapter.jar
BEAHOME/weblogic81/portal/lib/wsrp/adapters/wsrp-struts-adapter.jar
```

3. Redeploy the WSRP producer web application.

Aggressive Control Tree Persistence (SP4)

SP4 contains Portal Framework changes that significantly improve performance. Most of these changes have no effect on how you implement WebLogic Portal; however, because of the new aggressive control tree state management improvements, you must be particularly careful not to duplicate **session IDs** within your portal/desktop. If you use duplicate IDs, unknown effects might occur and the resulting problems would be difficult to diagnose.

Content Management Portlet (SP4)

The Content Management portlet now uses `wlp-admin.jar` rather than `cms_tools.jar`. If you added the Content Management portlet to your project in a previous Service Pack, you must delete `cms_tools.jar` and add the SP4 version of `wlp-admin.jar` to the following directory:

`<portal_application>\<project>\WEB-INF\lib\`

You can find the SP4 version of the `wlp-admin.jar` in the following directory:

`WL_HOME``\samples\portal\portalApp\sampleportal\WEB-INF\lib`

For more information on setting up a Content Management portlet, see the WebLogic Workshop online help.

URL Template File Updates (SP4)

The `WEB-INF\url-template-config.xml` file is automatically created in a portal Web project. The `url-template-config.xml` file contains multiple URL "templates," each with a unique name. Those template URLs contain variables such as `url:domain` and `url:port` that are read in from the active server.

If you are upgrading a web application from a previous service pack, review the following sections to determine if you need to edit the `url-template-config.xml` file.

Creating New Portals Using Administration Portal

When you create new portals with the Administration Portal, the generated `url-template-config.xml` file does not contain the `{url:currentPage}` variable. Add this variable to the file to allow the tree optimization feature to work, and to ensure full function of the Back button.

URL Template File Update for WSRP

The `url-template-config.xml` file contains URL templates and variables for remote portlets in a WSRP portal Web project. You must make the following two changes in this file for each WSRP URL template, so that the BEA-provided consumer will be able to handle mode changes and state changes from non-BEA producers:

- Replace `_mode` with `wsrp-mode`
- Replace `_state` with `wsrp-windowState`

Updates for Java Portlets

If you are using or you plan to use Java portlets (JSR 168 portlets), edit the `url-template-config.xml` file to match the new content in the SP4 version of this file; add/update the content as the following example shows:

```
<url-template name="portlet-secure-default">
https://{url:domain}:{url:securePort}/{url:path}?{url:queryString}{url:cur
rentPage}
</url-template>
<!-- Map Java portlet URLs to the templates defined above. -->
<java-portlet-url-templates>
<url-template-ref type="action" name="portlet-action"/>
<url-template-ref type="secure-action" name="portlet-secure-action"/>
<url-template-ref type="resource" name="portlet-resource"/>
<url-template-ref type="secure-resource" name="portlet-secure-resource"/>
<url-template-ref type="render" name="portlet-default"/>
<url-template-ref type="secure-render" name="portlet-secure-default"/>
</java-portlet-url-templates>
```

Portlet Preferences Behavior on Server Restart (SP4)

Prior to Service Pack 4, when the server was bounced or redeployed, the preferences in the `.portlet` file always overrode the preferences in the database, so administrator additions and changes to any portlet preferences were lost on a server restart.

A new option allows you to control the way in which portlet preferences are reloaded on server restart. You implement this control using the “master” attribute on the “propagate-preferences-on-deploy” element in the `WEB-INF/netuix-config.xml` file.

Possible values for the `master` attribute are:

file

Provides the same behavior as in previous versions. The preferences in the file system always take precedence. To preserve existing behavior, select this value.

database

In the case of a restart, the values in the database always takes precedence. In the case of a first time startup, the database is seeded from the `.portlet` files.

both

The default behavior if the attribute is missing. The `.portlet` preferences are merged with the database preferences, with the database values taking precedence over the `.portlet` file's values.

The following example shows the element with an attribute value of `both`:

```
<customization>
 <enable>true</enable>
 <propagate-preferences-on-deploy propagate-to-instances="true"
master="both"/>  <reload-database-on-redeploy reload="false"/>
</customization>
```

Entitlements Data Source (SP4)

The policy reference RDBMS persistence feature is a performance enhancement in SP4. If you are upgrading from a previous service pack and upgrading a pre-SP4 domain, you must add a new JNDI name, which is used to look up the p13nDataSource.

To perform this task using the WebLogic Server console:

1. In the Services Configurations section, select JDBC Data Sources.
2. In the Name column, select p13nDataSource.
3. Append `;p13n.entitlementsDataSource` to the `JNDI Name:` field and click Apply.
4. Restart the server.

Storage of Policy Reference Data in RDBMS (SP4)

Starting with WebLogic Portal 8.1 SP4, new database tables store entitlement data and Delegated Administration security policy reference data. Policy storage remains within LDAP, but policy reference data is stored in the RDBMS. When you access the Administration Portal for the first time in an upgraded domain, the system performs an automatic migration of this policy reference data.

If you want to prevent the migration of data to the new database tables, set the following Java environment parameter in the `startWebLogic.cmd/.sh` file:

```
-Dp13n.policyRef.LdapPersistence=Y
```

For details on the new database tables, see the Data Dictionary in the Portal Database Administration Guide:

http://edocs.bea.com/wlp/docs81/db/4Schemas.html

Using Role Caching with Entitlements (SP4)

Starting with WebLogic Portal 8.1 SP4, role values are cached automatically. Normally you do not need to change this setting. However, if you decide to define roles with expressions whose evaluation changes within the course of processing a request, you may need to disable this setting.

To disable role caching, edit the `web.xml` file for the respective application. For instructions on this task, see the WebLogic Portal Performance Tuning Guide at http://e-docs.bea.com/wlp/docs81/perftune/4PortalApplication.html#1074386.

Setting up JDBC Connection Polling (SP4)

In Service Pack 4, JDBC connection polling was added to the RDBMS authentication provider to keep JDBC connections active during long periods of inactivity. This was done to prevent firewalls or databases from timing out JDBC connections held by the internal pooling mechanism in the RDBMS authentication provider.

When configuring your RDBMS authentication provider you must enable connection polling. The WebLogic Server console includes the following two new attributes that you can set when you create/configure your provider:

- TestConnections - A checkbox used to enable/disable JDBC connection polling (disabled by default).
- TestInterval - The number of seconds between polling tests.

Portal Projects in Source Control (SP4)

BEA recommends managing your portal project using a source control system. For releases prior to SP4, BEA recommended not storing the class files in the path `*WEB-INF/classes*` because these are built automatically by WebLogic Workshop. However, in SP4 several new classes were added in the path `*WEB-INF/classes/com/bea/jsptools/portal*` for which source files are not provided, specifically:

- `PortalManager.class`
- `PortalLogger.class`
- `PortalBeanManager.class`
- `PortalAdminLogger.class`
- `DefaultPortalLogger.class`

Because of this change, SP4 portal projects based on WebLogic Workshop will not build successfully when extracted from their source control repository.

To ensure a successful build of your portal project while using source control, you can create a `.jar` file containing the class files listed above, and check the `.jar` file into the source control system repository.

Using New Proxy Portlet Cache (SP6)

In order to render each proxy portlet, WebLogic Portal makes several database calls to obtain proxy portlet information. WebLogic Portal 8.1 SP6 includes a new p13n cache called proxyPortletCache that allows you to cache the results of this getProxyPortlet query. BEA recommends that you use this cache if you are using federated portals (WSRP) and you have many proxy (remote) portlets; each environment is different, but an example might be an implementation with over 100 proxy portlets.

To customize the proxy portlet cache setting, edit the meta-inf/application-config.xml file and add the following entry:

```
<Cache MaxEntries="max_number_of_proxy_portlets" Name="proxyPortletCache"
Notes="This caches the ProxyPortlets from getProxyPortlet query"
TimeToLive="ttl_value"/>
```

For the value *max_number_of_proxy_portlets* enter a number, such as `150`.

For the value *ttl_value*, enter the timeout value in milliseconds; a value of `-1` causes the cache to never expire.

If you add this entry to the application-config.xml file, you can then configure or flush the cache using the WebLogic Portal Administration Console.

If you do not add this entry to the file, the cache will not be configurable using the Administration Console; the cache will still be used, with the following default values:

Time to live = 3600000 (one hour, in milliseconds)

Maximum entries = 100

Note: The proxy portlet cache will automatically be used only in new web applications. Even if you update an existing web application using the **File > Update Portal Libraries** selection, the proxy portlet cache will not be used unless you manually add the required entry to the application-config.xml file.

Using Content Query Expressions (SP6)

When you execute a Content Query expression in SP6, the order of precedence is not maintained and the expression generates incorrectly. The following expression (`a && (b || c)` is evaluated as (`a && b || c`).

This problem can be fixed in SP6 so that the order of precedence is maintained when you execute a Content Query expression. A system property was added so that the expression is evaluated the same as it was for 8.1 SP5 or earlier. Type the following to correctly generate a Content Query expression:

```
-Denable.content.rule.fix=true
```

Propagating Request Parameters from WSRP Requests (SP6)

Request parameters are no longer implicitly propagated from one WSRP request to the next. This issue impacts you if you have a patch for CR 254373. It affects all platforms.

Prior to the fix for CR254373, the portal framework would automatically add all request parameters to the WSRP `navigationState` URL. This action caused a problem because the URL could become too large. CR254373 removed this bug, but it caused pageflows to fail after the fix was applied. The steps involved included submitting a form to a pageflow action, the action then forwarded to a `result.jsp`, which then displayed the form contents by pulling them out of the request. This works in a standard pageflow but fails when the pageflow is accessed through a WSRP consumer portlet, because the submission of the form and the follow up display of the results requires two requests from the consumer to the producer (although it still appears to be a single request from the stand point a user interacting with the consumer). Submission of the form from the consumer to the producer is handled by a `performBlockingInteraction()` call. When that call completes, the consumer then follows up with a `getMarkup()` request to the producer that returns the markup for the `result.jsp` page. However, since the request parameters are not implicitly propagated from one request to the next, the form parameters are no longer present when the `getMarkup()` request is processed. The pageflow example described earlier is one way to demonstrate this issue, but it applies to any case where a `performBlockingInteraction()`call is made.

The workaround is not really a workaround; it is the proper way to implement this functionality. Implementers simply need to explicitly propagate the parameters they wish to keep in the request. Referring back to the pageflow example described above, there are two simple solutions that could be used to fix the implementation. In the first solution the pageflow action that receives the submitted form can simply pass it along by explicitly including the form in the Forward object that is returned by the action. This can be done using a netui ActionForm object, or individual parameter name value pairs can be included as pageInputs in the Forward object. The `results.jsp` should then retrieve the items in a manner suitable to the method used. Another possible solution is for the pageflow to hold onto the desired parameters as part of its state. That is the action that receives the form retrieves the parameters of interest from the ActionForm or request, assigns the values to instance variables and provides access to those value through getters on the pageflow. The `result.jsp` can then access the values need using those getters.

APPENDIX B

Changes to Visitor Tools for SP4

This appendix describes the functional and general code changes made to the Visitor Tools for SP4.

Functional Changes

The EditBook page now lets you add multiple books and pages as content. Previously, you had to select one book or page at a time.

Code Changes

This section lists specific JavaScript, JSP, Java, and other files that have been added to Visitor Tools or modified for SP4. The majority of work on Visitor Tools for SP4 has been to streamline the JSPs by relocating JavaScript to `.js` files and Java code to `.java` files.

In addition, the Visitor Tools code has been internationalized, and translatable strings have been placed in a `.properties` file.

The following sections describes these changes in detail.

Internationalization

The file `visitorTools/visitorTools.properties` was added to support internationalization. It contains translatable strings extracted from the code.

Moving JavaScript to .js Files

The following table lists the new JavaScript files that have been added to the `visitorTools/js` folder. The JavaScript code in these files was removed from the JSP files listed on the right.

New .js File	JavaScript Code Removed From:
`visitorTools/js/book.js`	`visitorTools/visitorToolsEditBook.jsp`
`visitorTools/js/main.js`	`visitorTools/visitorToolsMain.jsp`
`visitorTools/js/page.js`	`visitorTools/visitorToolsEditPage.jsp`

Changes to JSP Files

The following table lists JSP files that were modified along with the reason for the change.

Modified JSP	Reason for Change
`visitorTools/visitorToolsEditBook.jsp`	• Majority of JavaScript moved to `visitorTools/js/book.js` • Moved Java methods to `VisitorToolsUtil.java` and `VisitorToolsHtmlUtil.java` in `com/bea/jsptools/portal/util`
`visitorTools/visitorToolsEditPage.jsp`	• Majority of JavaScript moved to `visitorTools/js/page.js` • Moved Java methods to `VisitorToolsUtil.java` and `VisitorToolsHtmlUtil.java` in `com/bea/jsptools/portal/util`
`visitorTools/visitorToolsMain.jsp`	• Majority of JavaScript moved to `visitorTools/js/main.js` • Moved Java methods to `VisitorToolsUtil.java` and `VisitorToolsHtmlUtil.java` in `com/bea/jsptools/portal/util`

Changes to Java Files

This section lists the Java files that were added or changed for Visitor Tools along with the reason for each change.

com/bea/jsptools/portal/backing/VisitorToolsEditBookBacking.java
Moved some code out of the `handlePostbackData()` method to various distinct methods in same file.

com/bea/jsptools/portal/backing/VisitorToolsEditPageBacking.java
Moved some code out of the `handlePostbackData()` method to various distinct methods in same file.

com/bea/jsptools/portal/backing/VisitorToolsMainBacking.java
Moved some code out of the `handlePostbackData()` method to various distinct methods in same file.

com/bea/jsptools/portal/placement/BookPlacement.java
Added import optimization.

com/bea/jsptools/portal/placement/NavigablePlacement.java
Added import optimization.

com/bea/jsptools/portal/placement/PagePlacement.java
Added import optimization.

com/bea/jsptools/portal/placement/PlaceablePlacement.java
Added import optimization.

com/bea/jsptools/portal/placement/Placement.java
Added import optimization.

com/bea/jsptools/portal/placement/PortletPlacement.java
Added import optimization.

com/bea/jsptools/portal/PortalVisitorConstants.java
Added constants used by Visitor Tools code.

com/bea/jsptools/portal/PortalVisitorManager.java
Delegates to the PortalManager class for the majority of operations. This was done for code consolidation and ease of maintenance.

com/bea/jsptools/portal/util/EntitledPresentations.java
Added as a simple helper class.

com/bea/jsptools/portal/util/ResourceCount.java
Added as a simple helper class.

com/bea/jsptools/portal/util/VisitorToolsHtmlUtil.java

Added as a helper class for methods generating HTML. Contains mostly HTML generating methods pulled from `visitorToolsMain.jsp`, `visitorToolsEditPage.jsp`, and `visitorToolsEditBook.jsp`.

com/bea/jsptools/portal/util/VisitorToolsUtil.java

Added as a helper class for methods not generating HTML. Contains mostly methods pulled from `visitorToolsMain.jsp`, `visitorToolsEditPage.jsp`, and `visitorToolsEditBook.jsp`.

Unchanged Files

The following Visitor Tools files did not change for SP4:

`visitorTools/visitorTools.portion`

`visitorTools/js/dialog.js`

`visitorTools/skeletons/edittogglebutton.jsp`

APPENDIX C

Database Changes for SP4

This appendix includes details about WebLogic Portal Version 8.1 database schema changes for SP4.

Note: No database schema changes are required for Service Pack 5; however, you must update the default markup in the database, using the provided script, to be SP5 compliant. For instructions, see "Upgrading to Service Pack 5" on page 7-1.

- Entitlement Policy Reference
- Portal Framework

Entitlement Policy Reference

These new tables support the persistence of Portal entitlement and Delegated Administration security policy reference data. Policy storage remains within LDAP; however, now the policy reference data is stored in the RDBMS. When you access the Administration Portal for the first time in an upgraded domain, an automatic migration of this policy reference data is performed.

The following tables are new in SP4:

P13N_DELEGATED_HIERARCHY
P13N_ENTITLEMENT_APPLICATION
P13N_ENTITLEMENT_POLICY
P13N_ENTITLEMENT_RESOURCE
P13N_ENTITLEMENT_ROLE

Refer to the `.sql` files prefixed "`er_`" in the directory *`WL_HOME`*`/portal/db` for your DBMS to review the DDL for the Entitlement Policy Reference tables.

See the Data Dictionary for details about these new tables:

http://edocs.bea.com/wlp/docs81/db/4Schemas.html

Portal Framework

The following changes were made for each DBMS that WebLogic Portal supports.

- The PF_DESKTOP_DEFINITION has a new column named IS_TREE_OPTIMIZED. This column is a flag indicating whether tree optimization is on or off for a desktop. Acceptable values are 0 and 1, where 0=tree optimization is off (the default), and 1= tree optimization is on.

  Refer to the file named `81sp4_alter_tables.sql` under *`WL_HOME`*`/portal/db` for your DBMS to review the Database Definition Language (DDL) for this change.

- Several new indexes were created to improve performance; others indexes were dropped because they became redundant.

  Refer to the files named `81sp4_create_objects.sql` and `81sp4_drop_objects.sql` under *`WL_HOME`*`/portal/db` for your DBMS to review the DDL for this change.

Sybase Database Changes

The following Sybase-specific database changes occurred in SP4:

- `ON WEBLOGIC_INDEX` was added so that non-clustered indexes are placed onto their own segment and device; this change was made for administrative and data allocation/placement purposes, and performance is improved.

  See the `81sp4_drop_objects.sql` and `81sp4_create_objects.sql` files in *`WL_HOME`*`/portal/db/sybase/125` to review the DDL for this change.

- Sample scripts for "Configuring a Sybase Database" were modified to include the `WEBLOGIC_INDEX` device and segment definition.

  For details, see *`WL_HOME`*`\portal\db\sybase\125\admin`.

- Clustering indexes were reworked to improve performance.

  See the `81sp4_drop_objects.sql` and `81sp4_create_objects.sql` files in *`WL_HOME`*`/portal/db/sybase/125` to review the DDL for this change.

- `LOCK` clauses were added for each table to set either `DATAPAGES` or `DATAROWS` locking for each table. This excludes the tables for the Collaboration Portlets provided by Compoze.

See the files named `81sp4_lock_adj.sql` under *`WL_HOME`*`/portal/db/Sybase/125` for the DDL for the locking changes. Triggers are dropped and recreated as a result of these locking changes.

SQL Server Database Changes

The following SQL Server-specific database changes occurred in SP4.

- `ON WEBLOGIC_INDEX` was added so that non-clustered indexes are placed onto their own file group; this change was made for administrative and data allocation/placement purposes, and performance is improved.

  See the `81sp4_drop_objects.sql` and `81sp4_create_objects.sql` files in *`WL_HOME`*`/portal/db/sql_server/2000` to review the DDL for this change.

- Sample scripts for "Configuring a Microsoft SQL Server Database" were modified to include the `WEBLOGIC_INDEX` file group definition.

  For details, see *`WL_HOME`*`\portal\db\sql_server\2000\admin`.

- Clustering indexes were reworked to improve performance.

  See the `81sp4_drop_objects.sql` and `81sp4_create_objects.sql` files in *`WL_HOME`*`/portal/db/sql_server/2000` to review the DDL for this change.

DB2 Database Changes

The following SQL Server-specific database change occurred in SP4:

- Clustering indexes were reworked to improve performance.

  See the `81sp4_drop_objects.sql` and `81sp4_create_objects.sql` files in *`WL_HOME`*`/portal/db/db2/8` to review the DDL for this change

APPENDIX D

Database Changes for SP3

This appendix includes details about WebLogic Portal Version 8.1 database changes for SP3. For information on previous releases, consult your support representative as needed.

- Virtual Content Management
- Web Services for Remote Portlets (WSRP)
- WebLogic Commerce Services (CMV)
- Portal Framework

If desired, you can preview the changes that will be made to schema objects during the database upgrade to Service Pack 3. To do so, review the file `81sp3_alter_table.sql` for the DBMS in which you are interested. You can find this file in the specific folder for your specific DBMS and version in the path *`WL_HOME`*`\portal\db\<`*`dbms`*`>\<`*`version`*`>`. For example:

```
C:\bea813\weblogic81\portal\db\pointbase\44
```

Virtual Content Management

The following tables are new in SP3:

CMV_NODE
CMV_NODE_ASSIGNED_ROLE
CMV_NODE_VERSION
CMV_NODE_VERSION_PROPERTY
CMV_PROPERTY
CMV_VALUE

Refer to the `.sql` files prefixed "`cmv_`" in the directory *`WL_HOME`*`/portal/db` for your DBMS to review the DDL for the Virtual Content Management tables.

See the Data Dictionary for details about these new tables:

http://edocs.bea.com/wlp/docs81/db/4Schemas.html

For Virtual Content Management a new column named `LIFECYCLE_STATUS` is included in the CM_NODE table. `LIFECYCLE_STATUS` identifies the lifecycle status (for example, In Progress, Published, and so on) for this node. The default for new or existing nodes is 4, Published.

See the file named 81sp3_alter_tables.sql under *`WL_HOME`*`/portal/db` for your DBMS to view the DDL for this column addition.

Web Services for Remote Portlets (WSRP)

The following tables are new in SP3:

PF_CONSUMER_PORTLETS
PF_CONSUMER_PROPERTIES
PF_CONSUMER_REGISTRY
PF_PRODUCER_PROPERTIES
PF_PRODUCER_REGISTRY
PF_PROXY_PORTLET_INSTANCE

See the Data Dictionary for details about these new tables:

http://edocs.bea.com/wlp/docs81/db/4Schemas.html

See the `.sql` files prefixed "`wsrp_`" under *`WL_HOME`*`/portal/db` for your DBMS to view the DDL for the Web Services Remote Portals tables.

WebLogic Commerce Services (CMV)

The following changes occurred in SP3:

- In the tables WLCS_SAVED_ITEM_LIST and WLCS_SECURITY, some existing columns were changed to NOT NULL and Primary Keys were added.
- OLD_WLCS_SECURITY and OLD_WLCS_SAVED_ITEM_LIST tables were created for data retention and can be manually dropped following a successful upgrade.

See the file named `81sp3_alter_tables.sql` under *`WL_HOME`*`/portal/db` for your DBMS to review the DDL for the above changes.

Portal Framework

The following changes were made for each DBMS that WebLogic Portal supports.

- The following index was added: IX1_MARKUP_DEF ON PF_MARKUP_DEFINITION
- The following view was added: VIEW PF_PORTLET_HIERARCHY_V
- The following _LABEL columns were expanded from VARCHAR(40) to VARCHAR(80):
  - PF_BOOK_DEFINITION.BOOK_LABEL
  - PF_LOOK_AND_FEEL_DEFINITION.LOOK_FEEL_LABEL
  - PF_PAGE_DEFINITION.PAGE_LABEL

See the file named `81sp3_alter_tables.sql` under *`WL_HOME`*`/portal/db` for your DBMS to review the DDL for the above changes and additions.

APPENDIX E

Framework Reference for Portal Upgrades from 7.0 to 8.1

This appendix includes details about the WebLogic Portal framework, and is meant to aid developers in preparing applications developed in Portal 7.0 for re-creation in Version 8.1. This section includes information on the following topics:

- Security
- Portal Properties
- Framework File Reference
- JSP Reference

Security

This section explains some important aspects of security that should be considered when preparing to upgrade.

Security Framework JSPs

The security JSPs used in 7.0 do not have direct equivalents in 8.1.

Authentication Providers

This section explains how to authenticate across two authentication providers—Active Directory or RDBMS.

WebLogic 8.1 has a default security provider for Active Directory that can be used in place of the default LDAP provider used by WebLogic Server, WebLogic Portal, and WebLogic Platform in 8.1. For more information, consult the following resources:

- Choosing an Authentication Provider

  `http://edocs.bea.com/wls/docs81/secmanage/providers.html#1109511`

- Managing the Embedded LDAP Server

  `http://edocs.bea.com/wls/docs81/secmanage/ldap.html#1101845`

- Configuring an LDAP Authentication Provider

  `http://edocs.bea.com/wls/docs81/secmanage/providers.html#1172008`

Unified User Profile (UUP)

UUP is unchanged for WebLogic Portal Version 8.1. You can test this by compiling the dev2dev UUP example (for Portal 7.0) against the 8.1 JARs. For more information, see the UUP example at: `http://dev2dev.bea.com/codelibrary/code/unified_up.jsp`.

When building the example, you might see the following deprecation message about the DatabaseFactory:

```
src\examples\usermgmt\MyEntityPropertyManagerImpl.java:1073: warning:
getConnection(javax.sql.DataSource,int,long) in
com.bea.p13n.util.jdbc.DatabaseFactory has been deprecated
return DatabaseFactory.getConnection(getDataSource(), 3, 5);
```

The Javadoc on this error is as follows:

```
@deprecated Use DataSource.getConnection(), set retries and waitTime by
configuring the Pool
```

Also, make sure the classpath in the build script uses the new weblogic.jar, p13n_ejb.jar, and p13n_system.jar:

```
REM set these for your environment
SET BEA_HOME=C:\bea81beta
SET P13N_HOME=%BEA_HOME%\weblogic81b\p13n
SET WLSERVER_HOME=%BEA_HOME%\weblogic81b\server
SET JDK_HOME=%BEA_HOME%\jdk141_02
SET PACKAGE_DIR=examples\usermgmt

REM you shouldn't need to adjust these
SET J2EE_CLASSPATH=%WLSERVER_HOME%\lib\weblogic.jar
SET P13N_CLASSPATH=%P13N_HOME%\lib\p13n_ejb.jar;%P13N_HOME%\lib\p13n_system.jar
SET CLASSPATH=%P13N_CLASSPATH%;%J2EE_CLASSPATH%
```

Connecting to an Active Directory Server

WebLogic Portal 7.0 SP2 supports WLS Security Compatibility Mode, so connecting to an Active Directory server requires using the login framework included with that release.

The WLP 8.1 SP 3 LdapPropertyManagerImpl cannot use an 8.1 SP 2 `ejb-jar.xml` deployment descriptor. If you use the LdapPropertyManager UUP in `p13n_ejb.jar`, you must update the new `ejb-jar.xml` and `weblogic-ejb-jar.xml` deployment descriptors with your connection information.

If you try to use the old `ejb-jar.xml` deployment descriptor, then you will be missing the new `"useSSL" env-entry`. When this happens, the following exception occurs when you try to obtain user properties from LDAP:

```
javax.naming.NameNotFoundException: While trying to look up useSSL in
/app/ejb/p13n_ejb.jar#LdapPropertyManager/comp/env/config.; remaining name
'useSSL'
```

Portal Properties

As you modify a `.portal` or `.portlet` file in WebLogic Workshop Platform Edition, you can use the Property Editor window to set portal and portlet properties. This section provides reference material on all the editable properties belonging to portal elements.

Portal Component Properties

Table E-1 lists the properties that you can set for each portal component (desktop, book, page, portlet, and so on).

Note: The term "hint" in the descriptions means available capabilities that are not supported in the default skeletons provided with the WebLogic Workshop Portal Extensions.

Table E-1 Properties for All Portal Components

Administration Properties	**Markup Name**	Read-only. The unique name of a component markup type. For example, each of the three shell files (markup type "shell") must use a unique Markup Name inside their `.shell` files. (Because desktops, books, and pages do not have associated `.desktop`, `.book`, and `.page` files, an identical Markup Name is used for each.)

Table E-1 Properties for All Portal Components

Presentation Properties	**Presentation Class**	Optional. Typically a style sheet style, or class. Overrides the class attribute that would otherwise be used by the component's skeleton. For proper rendering, the class must exist in a cascading style sheet (CSS) in the desktop's selected skin, and the skin's `skin.properties` file must reference the CSS file. Sample - If you enter `my-custom-class`, the rendered HTML from the default skeletons will look like this: `<div class="my-custom-class">`
	Presentation ID	Optional. A unique ID inserted in the rendered HTML tag for the component. The value you enter (which must be unique among all presentation IDs in the portal) overrides the ID that might otherwise be inserted by the component's skeleton. An example use would be inserting a unique ID that JavaScript could operate on. Sample - If you enter `12345`, the rendered HTML from the default skeletons will look like this: `<div id="12345">`
	Presentation Style	Optional. HTML style attribute to insert for the portal component. This attribute is equivalent to a style sheet class attribute and overrides any attributes in the style sheet class. Separate multiple entries with a semicolon. Sample - If you enter `{background-color: #fff}` for a portlet title bar, the rendered HTML from the default skeletons will look like this: `<div class="bea-portal-window-titlebar" style="{background-color: #fff}">` and the portlet title bar will have a white background. The background color attribute you entered overrides the background color attribute in the `bea-portal-window-titlebar` class.
	Skeleton URI	Optional. The path (relative to the project) to a skeleton JSP that is used to render the portal component. This JSP overrides the skeleton JSP that would otherwise be used by the selected Look & Feel for the desktop. For example, enter `/frameworks/myskeletons/mytitlebar.jsp`.

Desktop Properties

Table E-2 lists the additional properties that appear when a desktop is selected in the Portal Designer.

Table E-2 Desktop Properties

Title	Required. Enter a title for the desktop. The title is used only for labeling in the Portal Designer. This is used at runtime for the HTMLTitle (bookmark name) when using a file-based portal. Change this in the shell file.
Look and Feel	Required. Select the Look & Feel to determine the default desktop appearance (combination of skins and skeletons). Change this in the shell file.
Shell	Required. Select the default shell for the area outside of the books, pages, and portlets. Shells determine the content for the desktop header and footer. Change this in the shell file.

Header and Footer Properties

Table E-3 lists the properties exposed when the Content node is selected under the Header or Footer node in the Portal Designer's Document Structure window. The following properties appear in the Property Editor window.

Table E-3 Header and Footer Properties

Content URI	Read-only. The JSP file referenced in a shell that is used to display content in the desktop header or footer. This value is read from the <netuix:jspContent> "contentUri" attribute in the `.shell` file `<header>` or `<footer>` tag. If you select a different shell for the desktop, you might see a different value here.
Error Page URI	Read-only. The file referenced in a shell that is used to display an error message in the desktop header or footer if the contentUri JSP encounters errors. This value is read from the <netuix:jspContent> "errorUri" attribute in the `.shell` file <header> or <footer> tag. If you select a different shell for the desktop, you might see a different value here.

Table E-3 Header and Footer Properties

Backing File	Read-only. The class referenced in a shell that is used for preprocessing prior to rendering a shell's header or footer JSP. This value is read from the <netuix:jspContent> "backingFile" attribute in the `.shell` file `<header>` or `<footer>` tag. If you select a different shell for the desktop, you might see a different value here.
Thread safe	Optional. Performance setting for books, pages and portlets that use backing files. When set to True, an instance of the backing file is shared among all books, pages and portlets that use the file. You can synchronize any instance variables that are not thread safe. When set to False, a new instance of the backing file is created each time the backing file is requested by books, pages and portlets that use the file.

Book and Page Properties

Table E-4 lists the properties exposed when a book or page is selected in the Portal Designer.

Table E-4 Book and Page Properties

Default Page (Book only)	Required. Select the page that appears by default when the desktop is accessed. The list is populated with Definition Labels of all pages in the portal.
Navigation (Book only)	Required. Select the default type of menu to use for navigation among books and pages.
Title	Required. Enter a title for the book or page. Page titles are used for the page tabs/navigation menus.
Theme	Optional. Select a theme to give the book or page a different look and feel from the rest of the desktop.
Definition Label	Required. Unique identifier for each book or page. A default value is entered automatically, but you can change the value. Each book or page must have a unique Definition Label. Definition Labels can be used to navigate to books or pages. Also, components must have Definition Labels for entitlements and delegated administration.
Backing File	Optional. If you want to use a class for preprocessing (for example, authentication) prior to rendering the book or page, enter the fully qualified name of that class. That class should implement the interface com.bea.netuix.servlets.controls.content.backing.JspBacking or extend com.bea.netuix.servlets.controls.content.backing.AbstractJspBacking.
Unselected Image	Optional. Select an image to specify the icon that appears next to the book or page title. This image appears on the tab of unselected pages.

Table E-4 Book and Page Properties

Selected Image	Optional. Select an image to specify the icon that appears next to the book or page title. This image appears on the tab of selected pages.
Rollover Image	Optional. Select a rollover image for the icon that appears next to the book or page title. This image appears when the mouse rolls over the tabs of unselected pages.
Orientation	Optional. Hint to the skeleton to position the navigation menu on the top, bottom, left, or right side of the book. You must build your own skeleton to support this property. Following are the numbers used in the .portal file for each orientation value: top=0, left=1, right=2, bottom=3.
Packed	Optional. Rendering hint that can be used by the skeleton to render the book or page in either expanded or packed mode. You must build your own skeleton to support the property. When packed is="false" (the default), the book or page takes up as much horizontal space as it can. When packed="true," the book or page takes up as little horizontal space as possible. From an HTML perspective, this property is most useful when the window is rendered using a table. When packed="false," the table's relative width would likely be set to "100%". When packed="true," the table width would likely remain unset.
Hidden	Optional. Hides the navigation tab for the book or page to prevent direct access. You can access the page or book with a link (to the definition label) or by using a backing file.
Layout Type (Pages only)	Required. Select the page layout style for positioning books and portlets in placeholders on a page.

Placeholder Properties

Table E-5 lists the properties exposed when a placeholder is selected in the Portal Designer. These are read-only properties, and they come from the `.layout` file that corresponds to the selected Layout Type for the page.

Table E-5 Placeholder Properties

Title	Read-only. The name of the placeholder. This value is read from the `.layout` file for the page's selected Layout Type.
Flow	Read-only. If the "Using Flow" property is set to true, this value can be "vertical" or "horizontal." Flow determines whether books or portlets put in the placeholder are positioned on top of each other (vertical) or beside each other (horizontal). This value is read from the `.layout` file for the page's selected Layout Type.
Using Flow	Read-only. If this value is set to "true", books and portlets put in the placeholder are positioned according to the value of the "Flow" property. If this value is set to "false", the default flow is used (vertical). This value is read from the `.layout` file for the page's selected Layout Type.
Placeholder Width	Read-only. Displays the width set for the placeholder. This value is read from the `.layout` file for the page's selected Layout Type.

Portlet Type Properties

Table E-6 lists the properties that can be set on Portlet Types. These are exposed when a portlet is opened in the Portlet Designer (as opposed to a Portlet Instance in the Portal Designer).

Table E-6 Portlet Type Properties

JSP Content (JSP Portlets Only)	**Content URI**	Required. The path (relative to the project) to the JSP or HTML file to be used for portlet's content. The Content URI was set when the portlet was created. You can use this property to point to a different file. For example, if the content is stored in `<project>/myportlets/my.jsp`, the Content URI is `/myportlets/my.jsp`.
	Error Page URI	Optional. The path (relative to the project) to the JSP or HTML file to be used for the portlet's error message if the main content cannot be rendered. For example, if the error page is stored in `<project>/myportlets/error.jsp`, the Content URI is `/myportlets/error.jsp`.
	Backing File	Optional. If you want to use a class for preprocessing (for example, authentication) prior to rendering the portlet, enter the fully qualified name of that class. That class should implement the interface com.bea.netuix.servlets.controls.content.backing.JspBacking or extend com.bea.netuix.servlets.controls.content.backing.AbstractJspBacking. See the Tutorial Portal in the Portal Samples for examples of backing files.
	Thread Safe	Optional. Performance setting for books, pages, and portlets that use backing files.When Thread Safe is set to "true," an instance of a backing file is shared among all books, pages, or portlets that request the backing file. You must synchronize any instance variables that are not thread safe. When Thread Safe is set to "false," a new instance of a backing file is created each time the backing file is requested by a different book, page, or portlet.
Page Flow Content (Page Flow Portlets only)	**Page Flow Action**	Optional. Identifies the initial action forwarded when this portlet calls its Page Flow.
	Listen To	Optional. For this portlet instance to respond to calls made by another Page Flow portlet, enter the portlet's instanceLabel in this field.

Table E-6 Portlet Type Properties

	Content URI	Required. Set this to the .JPF file that represents the Java Page Flow.
	Error Page URI	Optional. Designate an error page JSP in this field.

Table E-6 Portlet Type Properties

Portlet Properties	**Title**	Required. Enter the title that will appear in the portlet's title bar. You can override this title in each instance of the portlet.
	Orientation	Optional. Hint to the skeleton to position the portlet instance title bar on the top, bottom, left, or right side of the portlet. You must build your own skeleton to support this property in the `.portal` file. Following are the numbers used in the `.portal` file for each orientation value: top=0, left=1, right=2, bottom=3. You can override the orientation in each instance of the portlet.
	Packed	Optional. Rendering hint that can be used by the skeleton to render the portlet instances in either expanded or packed mode. You must build your own skeleton to support this property. When packed="false" (the default), the portlet takes up as much horizontal space as it can.When packed="true", the portlet takes up as little horizontal space as possible. From an HTML perspective, this property is most useful when the window is rendered using a table. When packed="false", the table's relative width would likely be set to "100%". When packed="true", the table width would likely remain unset.
	Definition Label	Required. Unique identifier for the portlet type. A default value is entered automatically, but you can change the value. Each portlet type must have a unique Definition Label. Definition Labels can be used to navigate to portlets. Also, components must have Definition Labels for entitlements and delegated administration.
	Default Minimized	Required. Select "true" for portlet instances of this type to be minimized when rendered. The default value is "false."
	Render Cacheable	Optional. To enhance performance, set to "true" to cache the portlet instances. For example, portlets that call Web services perform frequent, expensive processing. Caching Web service portlets greatly enhances performance. Do not set this to "true" if you are doing your own caching.
	Cache Expires (seconds)	Optional. When the "Render Cacheable" property is set to "true," enter the number of seconds in which the portlet instance cache expires.
	Fork Render	Optional. Intended for use by a portal administrator when configuring or tuning a portal. Setting to "true" indicates that the framework should attempt to multi-thread render the portlet. This property can be set to “true” only if the "Forkable" property is set to "true."
	Forkable	Optional. Allows a portlet developer to indicate whether or not the portlet is allowed to be multi-thread rendered. When set to "true," a portal administrator can use the "Fork Render" property to make the portlet multithread rendered.

Table E-6 Portlet Type Properties

Portlet Title Bar	**Icon URI**	Optional. The path (relative to the project) to the graphic to be used in the portlet title bar. You must create a skeleton to support this property.
	Help URI	Optional. The path (relative to the project) to the portlet's help file.
	Edit URI	Optional. The path (relative to the project) to the portlet's edit page.
	Can Maximize	Optional. If set to "true," the portlet can be maximized.
	Can Minimize	Optional. If set to “true,” the portlet can be minimized.
	Can Delete	Optional. If set to "true," the portlet can be deleted from a page.

Portlet Instance Properties

Table E-7 lists the subset of Portlet Type Properties exposed when a portlet instance is selected in the Portal Designer (as opposed to the `.portlet` source file).

Table E-7 Portlet Instance Properties

Title	Required. Enter a title only if you want to override the default title provided by the `.portlet` file. The title is used in the portlet title bar.
Instance Label	Required. A single portlet, represented by a `.portlet` file, can be used multiple times in a portal. Each use of that portlet is a portlet instance, and each portlet instance must have a unique ID, or Instance Label. A default value is entered automatically, but you can change the value. Instance Labels are necessary for inter-portlet communication between Java Page Flow portlets. Also, portlets must have Instance labels for entitlements and delegated administration.
Portlet URI	Required. The path (relative to the project) of the parent `.portlet` file. For example, if the file is stored in `<project>\myportlets\my.portlet`, the Portlet URI is `/myportlets/my.portlet`.

Table E-7 Portlet Instance Properties

Theme	Optional. Select a theme to give the portlet a different look and feel from the rest of the desktop.
Orientation	Optional. Hint to the skeleton to position the portlet title bar on the top, bottom, left, or right side of the portlet. You must build your own skeleton to support this property. Following are the default values used in the `.portal` file for each orientation value: top=0, left=1, right=2, bottom=3. Change the value for this property only if you want to override the default orientation provided by the `.portlet` file.
Default Minimized	Optional. Select "true" for the portlet to be minimized when it is rendered. The default value is "false." Change the value for this property only if you want to override the default value provided by the `.portlet` file.

Framework File Reference

This section includes tables describing several framework files that have been re-located, replaced or created in WebLogic Portal Version 8.1.

Framework Files

Table E-8 lists commonly edited framework files and their new locations, replacement files, and so on.

Table E-8 Portal Framework File location changes from 7.0 to 8.1

7.0 in **/framework**	8.1 in **/framework/skeletons/<skeletonname>**	Notes
edit.jsp	window.jsp	Portlet modes handled by framework
edit_titlebar.inc	titlebar.jsp	title bar variants handled by framework
footer.jsp	footer.jsp	Portlet states handled by framework
header.jsp	header.jsp	
header_links.inc	n/a	
hnav_bar.jsp	singlelevelmenu.jsp	
hportal.inc	desktop.jsp	

Table E-8 Portal Framework File location changes from 7.0 to 8.1

7.0 in **/framework**	8.1 in **/framework/skeletons/<skeletonname>**	Notes
maximize.jsp	window.jsp	Portlet states handled by framework
maximize_titlebar.inc	titlebar.jsp	Portlet states handled by framework
minimize_titlebar.inc	titlebar.jsp	Portlet states handled by framework
normal_titlebar.inc	titlebar.jsp	Portlet states handled by framework
page.jsp	page.jsp	
portal.jsp	desktop.jsp	
resourceURL.inc	n/a	
titlebar.jsp	titlebar.jsp	
vnav_bar.jsp	singlelevelmenu.jsp	
vportal.inc	desktop.jsp	
n/a	book.jsp	New in release 8.1
n/a	borderlayout.jsp	
error.jsp	error.jsp	
see Layouts	flowlayout.jsp	
see Layouts	gridlayout.jsp	
portal.jsp	head.jsp	
n/a	multilevelmenu.jsp	New in release 8.1
see Layouts	placeholder.jsp	
n/a	shell.jsp	New in release 8.1
n/a	submenu.jsp	New in release 8.1

Table E-8 Portal Framework File location changes from 7.0 to 8.1

7.0 in **/framework**	8.1 in **/framework/skeletons/<skeletonname>**	Notes
n/a	theme.jsp	New in release 8.1
n/a	togglebutton.jsp	
n/a	togglebuttondelete.jsp	

JSP Reference

This section includes tables meant to ease the process of upgrading existing applications by providing an extremely simplistic correspondence between JSP usage from WebLogic Portal 7.0 SP2 to WebLogic Portal 8.1. For detailed information on using JSP tags, consult the Javadoc at the following link:

`http://edocs.bea.com/workshop/docs81/doc/en/portal/buildportals/navReference.html`

JSP Tag Changes

Table E-9 lists changes to the supported JSP tags from WebLogic Portal 7.0 SP2.

Table E-9 JSP Tag Libraries in WebLogic Portal 7.0 SP2

Tag Library	Used in Task	Changes in WebLogic Portal 8.1
ad.tld	Ad placeholders	None.
catalog.tld	Catalog Service Management	Must be installed explicitly in Workshop.
cm.tld	Content Management	Significant changes.
eb.tld	E-business Service Management	None.
es.tld	Personalization Utilities	es:notNull has been changed to consider an empty string to be “null.” Formerly, es:isNull and es:notNull behaved inconsistently for empty (zero-length) strings. If you were relying on the inconsistent empty-string behavior, refactor JSPs that use the notNull tag.
i18n.tld	Internationalization Management	Three tags added.

Table E-9 JSP Tag Libraries in WebLogic Portal 7.0 SP2

Tag Library	Used in Task	Changes in WebLogic Portal 8.1
ph.tld	Placeholder Management	None.
portal.tld	Portal Management	Deprecated. See Portal Skeleton Rendering.
portlet.tld	Portlet Management	Deprecated. See Portlet Preferences.
productTracking.tld	Event and Behavior Tracking Management	Significant changes.
ps.tld	Property Set Management	None.
pz.tld	Personalization Management	Significant changes. Called "User Segment Rule Display" in WebLogic Workshop.
tracking.tld	Event and Behavior Tracking Management	None.
um.tld	User/Group Management	None.
util.tld	Portal/Portlet Management	None.
webflow.tld	Navigation Management	Deprecated. See Page Flow.
wl	WebLogic Utilities	None.

JSP Tag Libraries

Table E-10 lists changes to the supported JSP tag libraries in WebLogic Portal Extensions.

Table E-10 JSP Tag Libraries

WebLogic Portal 7.0 SP2	Changes	WebLogic Portal 8.1
ad.tld	None.	ad-taglib.jar
cat.tld	None.	cat-taglib.jar

Table E-10 JSP Tag Libraries

WebLogic Portal 7.0 SP2	Changes	WebLogic Portal 8.1
cm.tld	Significant. See Table E-11.	• cm-taglib.jar and pz-taglib.jar are replaced by content.tld and content.jar and a new version of pz-taglib.jar • cm-taglib.jar and pz-compat.taglib.jar are available in Portal Compatibility Domain
	New. Multichannel tags.	client_taglib.jar
eb.tld		
es.tld		
i18n.tld	Three tags added.	
ph.tld		
portal.tld		Supported only in Portal Compatibility Domain.
	New	portalinterop-taglib.jar
	New	portletinterop_taglib.jar
portlet.tld		portlet_taglib.jar
	New in release 8.1.	Portal Interface Rendering
	New in release 8.1.	Portlet Preferences
productTracking.tld		
ps.tld		
pz.tld	Significant. See Table E-11.	
	New in release 8.1.	render-taglib.jar provides skeleton rendering tags
tracking.tld		
um.tld		
util.tld		

Table E-10 JSP Tag Libraries

WebLogic Portal 7.0 SP2	Changes	WebLogic Portal 8.1
webflow.tld	Deprecated	
wl	WebLogic Utilities	

Notes on Individual JSP Tags

Table E-11 includes specific JSP tags and some changes in the Version 8.1.

Table E-11 Portal JSP Tag Changes/Replacements

WebLogic Portal 7.0 SP2	WebLogic Portal 8.1	Notes
um:tld • GetProperty • PrintProperty • PrintDoc	GetProperty	Retrieves a property value from a content node and stores it as a variable or prints it in the JSP.
SelectByID	GetNode	Retrieves a content node based on an explicit path and stores it in a variable.
Select	Search	Searches for and retrieves content nodes based on a supplied query and stores the results in a variable.
ContentSelector	ContentSelector (returns Node instead of Content Object)	Implements a Content Selector defined with the WebLogic Workshop Portal Extensions. Displays personalized Web content in a JSP based on Content Selector rules and queries defined with the WebLogic Workshop Portal Extensions.
ContentQuery	ContentQuery (returns Node instead of Content Object)	Performs a content attribute search in a content management system and returns an array of content objects.
n/a	include	Includes a localized version of the page.
n/a	forward	Forwards a localized version of the page.
n/a	resolve	Resolves to a localized version of the page.

Table E-11 Portal JSP Tag Changes/Replacements

WebLogic Portal 7.0 SP2	WebLogic Portal 8.1	Notes
n/a	adTarget	Uses the Ad Service to send an ad query to the content management system.
n/a	render	Renders a content node from the Virtual Content Repository in the current JSP.
n/a	beginRender	Used in the portal skeletons for rendering a portal resource. This tag defines the opening HTML tag for a resource.
n/a	createId	Creates an ID for a rendered component.
n/a	CreateUri	Creates a URI based on the application's configured skin path.
n/a	endRender	Used in the portal skeletons for rendering a portal resource. This tag defines the closing HTML tag for a resource.
n/a	jspContentUrl	Lets you create URLs to windows and set the content of those windows.
n/a	pageUrl	Lets you create links to switch to a page or a book based on labels.
n/a	jspUri	Retrieves the location of the JSP in which this tag is used.
n/a	param	Lets you append query parameters to the URL tags.
n/a	renderChild	Used in the portal skeletons for rendering a portal resource. This tag is used to render portlet title bars, title bar buttons, menus (navigation such as page tabs), and table cells within a layout.
n/a	toggleButtonUrl	Used in the portal skeletons for rendering a portal resource. This tag lets you build URLs for the toggle buttons in portlet title bars.
n/a	windowUrl	Lets you create links to windows based on labels and switch a window to a mode, state, or both.
n/a	writeAttribute	Used in the portal skeletons for rendering a portal resource. This tag sets HTML attributes for a tag.
n/a	writeId	Writes an ID for a rendered component.

Table E-11 Portal JSP Tag Changes/Replacements

WebLogic Portal 7.0 SP2	WebLogic Portal 8.1	Notes
n/a	writeUri	Writes the URI based on the application's configured skin path.
n/a	getPreference	Gets the default value of a portlet preference or sets a default value.
n/a	getPreferences	Gets all the values of a given portlet preference, or supplies a default value.
n/a	forEachPreference	Iterates over all the preferences available for a portlet and stores the results in a variable.
n/a	ifModifiable	Includes the body of the tag if the given portlet preference is modifiable.
n/a	else	Includes the body of the tag if the given portlet preference is not modifiable.
n/a	default	Renders its contents only if the client's classification has been mapped to "default" or if the client is not recognized.
n/a	not-default	Renders its content only if the client classification has been mapped to anything other than "default."
n/a	recognized	Renders its content if the client has been mapped to any classification, even "default."
n/a	not-recognized	Renders its content if the client has not been mapped to any classification.
n/a	when	Renders its content for any client classification listed in this tag's "client" attribute that matches the client classification name in the HTTP request.
n/a	when-not	Renders its content for any client classification not listed in this tag's "client" attribute. The client name comes from the HTTP request.