BEA WebLogic Portal®

Upgrade Guide

Version 8.1
July 2003
Revised: July 2003

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

About This Document

Upgrading WebLogic Portal Applications

Compatibility Domain

Upgrading to WebLogic Portal 8.1

Applying Service Packs

Database and Metadata Upgrade Steps

A. Portal Framework Details

About This Document

This document explains procedures for upgrading WebLogic Portal applications built on version 7.0 SP2 to run on release 8.1.

This document covers the following topics:

- Chapter 1, "Upgrading WebLogic Portal Applications" gives an overview of the procedures and elements to be considered when undertaking any upgrade project.
- Chapter 2, "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, suggesting helpful implementation considerations where possible.
- Chapter 5, "Database and Metadata Upgrade Steps" explains the process by which WebLogic Portal 7.0 SP2 application data is converted for use in 8.1 Portal applications.
- Appendix A, "Portal Framework Details" includes details about the WebLogic Portal framework meant to aid developers in preparing existing applications for re-creation in the newest version of the product.

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.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on 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 you want to print.

If you do not have the Adobe Acrobat Reader, you can get 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/

For more information about CORBA and distributed object computing, transaction processing, and Java, refer to the Bibliography at http://edocs.bea.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.

Convention	Item

`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`** `( )`
`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.

Convention	Item
. . .	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

Upgrading WebLogic Portal Applications

Overview

This document explains the strategies and procedures for upgrading WebLogic Portal 7.0 SP2 applications to WebLogic Portal 8.1. To begin with, basic strategies for upgrading existing Portal applications are introduced, along with recommendations and examples.

The following topics are introduced in this section:

- The Upgrade Process
- Comparing Supported Features
- How to Upgrade Existing Applications

The Upgrade Process

There are two paths for upgrading your WebLogic Portal 7.0 SP2 portal web application to 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 the Hosting 7.0 Applications in 8.1 Compatibility Domain section.

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

Hosting an existing Portal Web application in a WebLogic Portal 8.1 domain requires following the procedure outlined in the Upgrading from Compatibility Mode to WebLogic Portal 8.1 section.

Definitions

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

Migration

Moving an application/domain from a third party technology to a BEA product. e.g. migrating a customer from IBM, webMethods or "home grown" to BEA.

Upgrade

Updating BEA platform (and components) from older release/SP to newer release/SP, this includes updating existing application/domain to run in newer version, e.g. 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 via standard protocols.

Compatibility

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

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 A, "Portal Framework Details".

The portal framework in 7.0 has a one-to-one mapping with the portal it represents, which is in-line with 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.

Note: This section provides a summary of feature and framework changes general to the WebLogic Portal 8.1 platform. For detailed information on each of these issues, consult Appendix A, "Portal Framework Details".

In WebLogic 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 may 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 a .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 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, etc. 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, etc.) 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, etc. 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 may 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 may 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, etc. 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 pla ce elements on a grid, at the center, north, south, east, or west, or in a horizontal or vertical flow, respectively.

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 will be 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. These may be modified 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 may allow portlets to span rows or columns.

Group Portals

Group portal definitions will not be 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 Pageflow 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 Pageflow.

Exceptions

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

Pageflows

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 Pageflow 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. Additionaly, 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 Upgrading from Compatibility.

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 Upgrading Existing Behavior Tracking Data.

Struts

Some customers may have existing struts applications or desire to start developing Struts applications on WebLogic Portal 7.0 because of the move to Pageflows 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.

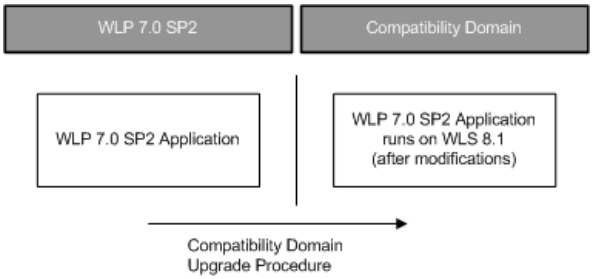
How to Upgrade Existing Applications

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, explained in the "Hosting 7.0 Applications in 8.1 Compatibility Domain" section, or to upgrade the application to run in WebLogic Portal 8.1, explained in the "Upgrading from Compatibility Mode to WebLogic Portal 8.1" section. The bulk of this guide is devoted to the Portal Compatibility Domain path, but also includes significant information required for re-implementation for 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.

Figure 1-1 Upgrade to Compatibility Domain



The capabilities and limitations of this configuration are explained here:

Features and Limitations of the Compatibility Domain

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

Generally speaking, WebLogic 7.0 portals running in a Portal Compatibility Domain can be administered using the 7.0 Weblogic Portal Administration Tools. WebLogic Portal 8.1 features are generally not supported in a portal enterprise application running in the Portal Compatibility Domain. Development support is also limited. For instance, neither the E-Business Control Center nor WebLogic Workshop is supported for use with any application once it has been upgraded to run in a Portal Compatibility Domain.

- 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 seqencer must have its own datasource. So all the APIs where you could pass one in have been removed. For information on replacing Upgrading from Compatibility Mode to WebLogic Portal 8.1 section of this document.

- MBean Configuration Mechanism

  The configuration of Portal MBeans (mail service, campaign service, ad service, behavior tracking, events, caches, etc) has changed: In a Portal Compatibility Domain, these settings must be changed by editing the `<ent-app>/META-INF/application-config.xml` file and re-starting the server.

- Using Existing Campaigns

If you are using existing campaigns in a compatibility domain, use the content JSP tags from the previous release. If you are creating new campaigns or using existing campaigns in a new domain, use the latest content JSP tags.

Runtime and Administrative Issues

This section explains issues concerning running and adminstering applications in Compatibility Domain, and discusses differences between 7.0 and 8.1 applications.

Portal Compatibility Domain supports all runtime features of the WLP 7.0 SP2 with the following exceptions:

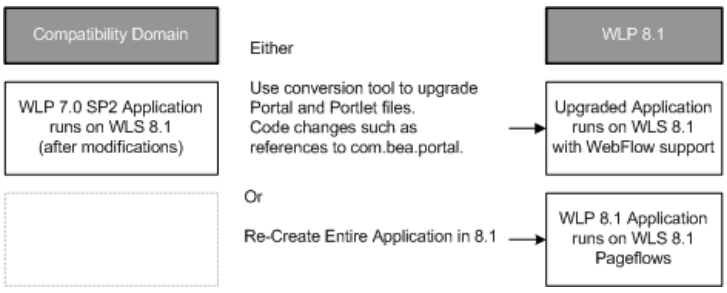
- In the WebLogic Portal Administration Tools, Payment Management will not be available.
- FileRealm users and groups will not be automatically available.

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) will work with a portal. The net difference is that the fileRealm users and groups will no longer be automatically available in a Compatibility Domain. These are typically system users and groups.

Upgrading to Weblogic Portal 8.1

Upgrading from Compatibility Mode to WebLogic Portal 8.1 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 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 a command line tool that converts .portal and .portlet files from version 7.0 SP2 to version 8.1.

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 listed in the Upgrading from Compatibility Mode to WebLogic Portal 8.1 section.

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 8.1 release. For details on Look and Feel implementation, consult the Presentation Framework section.

CHAPTER 2

Compatibility Domain

Hosting 7.0 Applications in 8.1 Compatibility Domain

WebLogic Portal 8.1 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 build on WebLogic Portal 7.0 SP4.

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 may 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 seqencer must have its own datasource. So all the APIs where you could pass one in have been removed. The Framework File Reference section 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 `ENT-APP` 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 the 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) will work with a portal. 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 may 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 only see 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 would require simply installing the `RDBMSAuthenticationProvider` and designating it 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:

- Creating a Portal Compatibility Domain
- Upgrading the Enterprise Application
- 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 via 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 jars 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.

Backup 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 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 steps that follow 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. Listing 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:9093/wlportal"/>
```

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 Schema Upgrade

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

   Follow the steps in section Upgrade PointBase Triggers. These steps prepare the PointBase files in upgrade/pointbase to be schema-upgraded.

4. Upgrade 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 the

Upgrade WebLogic Portal 7.0 Schema section of Chapter 5, "Database and Metadata Upgrade Steps".

Upgrade Existing Data

This step causes the contents of the ENTITLEMENT_RULESET table to be updated.

All Databases

Follow the steps in the section Upgrade ENTITLEMENT_RULESET Records section of Chapter 5, "Database and Metadata Upgrade Steps".

Specifically, the following changes are made by 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 the security. The following tasks are included in this section:

- Install an RDBMSAuthenticationProvider to access existing users and groups stored in the RDBMS.
- Any portal users and groups previously in **fileRealm.properties** required for access for the portal running in portal compatibility will need to be added to the RDBMS and their passwords reset.
- The PortalSystemAdministrators group is required by portal compatibility (and WebLogic Portal 8.1 in general) in order to make changes in the WebLogic Portal Administration Tools, so this group is manually added to the RDBMS.
- Users with portal administrator privileges in the existing WebLogic Portal 7.0 domain are added to the PortalSystemAdministrators group.

- Any ACLs configured in the WebLogic Portal 7.0 **`fileRealm.properties`** (or on an LDAP store) must be reconfigured: 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 this section if your 7.0 domain did not use the RDBMSRealm. If you are unsure of 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 this section. 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 the steps 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 may need to adjust the host name and port to match the settings in `config.xml`.
2. Login as **weblogic/weblogic**.Navigate to **Security-->Realms-->myrealm-->Providers-->** and select **Authentication**.
3. Click **Configure a new `RDBMSAuthenticator`** and 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 in Listing 2-5, and using sampleportal with Oracle, the settings would be as in Listing 2-6.
5. Click **Apply.**
6. Go to **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, the PortalSystemAdministrators group must be added to the `RDBMSAuthenticationProvider`. If users and groups were stored in the database in the existing WebLogic Portal 7.0 domain, the PortalSystemAdministrators group must be added to RDBMSAuthentication.

Using the SQL editor of your choice (for example, SQL*Plus, 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 section 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 may 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 section below, `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.

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 (i.e., 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 `<8.1 domain>\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 objects into 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** 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 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 in Listing 2-8. If this is the case, take the steps in the section Step 1: Change the Content Management application-config.xml File.

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.

Note: 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 get 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 only present 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:\gibi12\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:\gibi12\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:\gibi12\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:\gibi12\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:\gibi15\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 may 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 we simply compile it and stick it in `wlcsSample.jar`. Therefore, a proxy jar file must be generated, then any changed proxy files must be manually replaced.

Here is an example of a target which generates proxy jar files -- add an entry such as 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-Synch Files

Follow the 'upgrade application-sync' steps in the Upgrade Application-Sync Files section.

Final Adjustments to the Domain

This set of steps requires at least one enterprise application to be upgraded and deployed, since 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 userid 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, go to Security -> Realms -> myRealm -> Global Roles, and click on "Configure a new Global role".
   - administrator
   - demosa1
   - demosa2
6. 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:

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. Shutdown the server

Users and Groups Specified in fileRealm.properties

Any system-level users and groups which under 7x were specified in fileRealm.properties must be added to the authentication provider used by WebLogic Portal 8.1.

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, etc) 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 that you wish to provide portal access and portal tool access in the new Portal Compatibility Domain.

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

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 pertinant to upgrade issues. These aspects of the new WLP framework include Webflow support, Portal services, Content Management and Struts support.

Upgrading from Compatibility

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

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, consult the Procedure for Hosting in Compatibility.

1. Run the 'upgradePortalFiles.cmd' utility and place you 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. You must then make sure the jsps referenced by the portlets are in the correct relative location also.

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 on 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 chose **Portal Web Project** then browse to the web application. Make sure the "Copy into Application directory" option is checked. You will be prompted with a message saying "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 which are not supported in WebLogic Portal 8.1. This will likely mean that other area of your application will need to be re-written.
6. If the application uses pipeline components, support for this functionality needs to be added. Right-click on 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: Right-click on **Modules** in the Application pane, select **Add Module...** browse for the files, and click **Open**.

Note: You should not need the **tools700.war** or the **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 Compatability 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 on the data directory in the Workshop Application pane and select **Import...** Then browse to the directory from the Portal Compatibility Domain and and click **Import**.
   - To merge individual files into an existing data sub-directory, (campaigns, for example) right-click on the Workshop Application pane and select **Import...** Then browse to the directory from the Portal Compatibility Domain and 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**, which may need to be created.
    - This placement is not required for the the application to function, however, the upgradePortalFiles tool assumes this relative location. Therefore, if you choose a different organization, you may 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 fix up the layout as the upgrade tool chooses a simple single column layout that may not be suitable for you 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 an additions to the server classpath that may be required by your application.
16. If errors are being generated, you may need to search for references to com.bea.portal classes. These need to be replaced with references to new classes.
17. The next step would be to replace webflows with pageflows and the WebflowServlets 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 elective.

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.

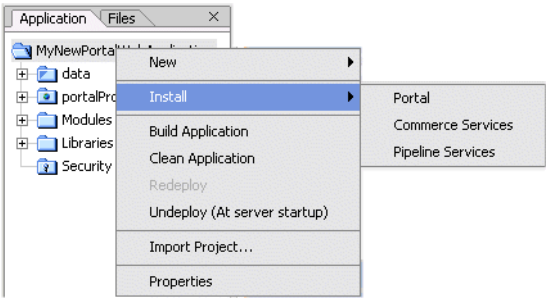
1. Copy the tools700.war into the J2EE Application root folder.

   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 will be deployed automatically.

   Note: The location, as well as the Web Application name, are changed in the new release of WebLogic Portal.

2. If commerce services haven't been installed in the application, right-click on 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. **tools700** is the default.

4. To use these tools, open a Web brower 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. 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 suych as Tiles.

Support for Struts in WebLogic Portal 8.1

Struts applications configuration files can be edited in WebLogic Workshop in 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 Pageflow. These updates are merged to the struts-config XML file generated from the Java Pageflow (.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 Pageflows in WebLogic Portal 8.1

Pageflow 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 Pageflow will require some changes to JSP tags in order to take advantage of Pageflow features.

JSP Tag Replacements

Appendix A, "Portal Framework Details" details JSP tags deprecated in WebLogic Portal 8.1, suggesting replacement tags or other refactoring strategies where possible.

Applying Service Packs

This topic provides instructions for applying service pack changes to your portal applications after you install the service packs. For instructions on installing service packs, see Installing Service Packs and Rolling Patches in the WebLogic Platform documentation at http://e-docs.bea.com/platform/docs81/install/update.html.

Updating Portal Libraries with New Service Packs

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. 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 you want to update.
3. In the Application window, right-click the application directory and choose **Install > Update Portal Libraries**.
4. If the service pack includes Commerce or Pipeline updates, 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 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 the service pack includes updates to 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**.

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

   If you have the WebLogic Administration Portal (`<portalApplication>/Modules/adminPortal.war`) hidden, use your operating system file manager to copy the new `adminPortal.war` from the `<WEBLOGIC_HOME>/portal/lib` directory into the root directory of the your portal applications in which this file is hidden.

7. Restart your server.

8. 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 Deploying Portal Applications for deployment instructions.

CHAPTER 5

Database and Metadata Upgrade Steps

Overview

This document details several upgrade procedures for WebLogic Portal database and meta-data files. Table 5 -1 explains when each procedure is used.

Table 5 -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

The following topics are included in this section:

1. Upgrade Pointbase Triggers
2. Upgrade WebLogic Portal 7.0 Schema
3. Upgrade Application-Sync Files
4. Upgrade Portal and Portlet Files

5. Upgrade ENTITLEMENT_RULESET Records
6. Upgrading Existing Behavior Tracking Data

Database Upgrade Overview

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

Upgrade Pointbase Triggers

With the release of WebLogic 8.1, a more current release of PointBase (4.4) is being shipped. PointBase rewrote their implementation of database triggers in PointBase 4.3. As a result of this, you must execute the following steps to upgrade your PointBase database for use in a 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 for 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 going to "***SQL -> Execute All***":

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

5. Shutdown 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. Shutdown the PointBase console and Pointbase server.

Upgrade WebLogic Portal 7.0 Schema

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

1. Backup your 7.0 database
2. Change directories to ..\weblogic81\portal\upgrade
3. Update db_settings.properties with your 7.0 database settings
4. Execute the following script: upgrade_db_schema_to_81
5. Review upgrade_db_schema_to_81.log to verify that the upgrade was successful.

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

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

The database is now upgraded to WebLogic Portal 8.1.

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 wlpCompatability domain you must rename the *$1.wal and *.dbn files to match the PointBase database name used in your domain.

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, execute 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 will be 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 the ..\META-INF\data already exists you will be 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. Execute the upgrade script

   UpgradeApplicationSyncFiles <

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

Review the corresponding log file (upgradeApplicationSyncFiles.log) to verify that the upgrade was successful. The same information is displayed to the console.

Upgrade Portal and Portlet Files

Upgrade .portal and .portlet files to make them using the script in the following directory:

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

From within the upgrade directory, (..\weblogic81\portal\upgrade) execute the script by typing the following at the command line:

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

For example, Listing 5-1 illustrates the output if the **sampleportal-project** directory were copied into a directory called **`C:\old`** and a directory called **`C:\new`** were created for the transformed portal and portlet files.

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

Listing 5-1 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 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 may 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" will be created and any portals found in the input directory are written here after they are transformed if they pass 8.1 validation. Under the "portal" directory a directory named "portlets" is created and all portlets found in the input directory are written here after they are transformed

and they pass 8.1 validation. 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. Note that none of the source files are changed or moved.

The output directory may look 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)
```

Upgrade ENTITLEMENT_RULESET Records

Entitlements have changed in 8.1. This upgrade step is only necessary if you are planning on running in compatibility mode.

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

For compatibility mode, upgrade ENTITLEMENT_RULESET records in the database using the following steps:

1. Backup the 7.0 database.
2. For PointBase only: Start the PointBase Server by running: **`startPBServer`** from the `..\bea81\weblogic81\portal\upgrade` directory.

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

   ```
   -port=9095
   ```

3. Change directories to: **`..\bea81\weblogic81\portal\upgrade`**.
4. Update upgrade.properties with your 7.0 database settings.
5. Execute the following script: `upgradeEntitlementRulesets`.

6. Be sure and review the upgradeEntitlementRulesets.log to verify that the upgrade was successful. The same information is displayed to the console.
7. For PointBase; Shutdown the PointBase Server

Upgrading Existing Behavior Tracking Data

To preserve and reuse existing behavior tracking data in an upgraded application, take 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

APPENDIX A

Portal Framework Details

This appendix includes details about the WebLogic Portal framework meant to aid developers in preparing existing applications for re-creation in the newest version of the product. This section includes information on the following topics:

- Security
- Presentation Framework
- Framework File 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 security provider.

Weblogic 8.1 has a default security provider for Active Directory which can be used in place of the default LDAP provider that is used by WLS, WLP and the Platform in 8.1. By contrast, 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. 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 a LDAP Authentication Provider

  http://edocs.bea.com/wls/docs81/secmanage/providers.html#1172008

Unified User Profile

UUP is unchanged for 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%
```

Presentation Framework

WebLogic Portal 8.1 introduces a powerful and flexible presentation framework for portals, enabling an unprecedented degree of customization and standardization. This section explains the presentation framework, providing procedures for creating new presentation elements.

Look and Feel Architecture

A portal Look and Feel determines the physical appearance of a portal. WebLogic Portal's Look and Feel architecture provides flexible, powerful framework for determining the appearance of your portals. This section introduces the parts of a Look and Feel object - namely, Skins, Skeletons, and Themes - and explains the relationships among them.

Note: For instructions on creating custom Look and Feel elements, layouts and navigation menues, consult the Working with Look and Feel Elements section of the WebLogic Portal 8.1 Best Practices Guide.

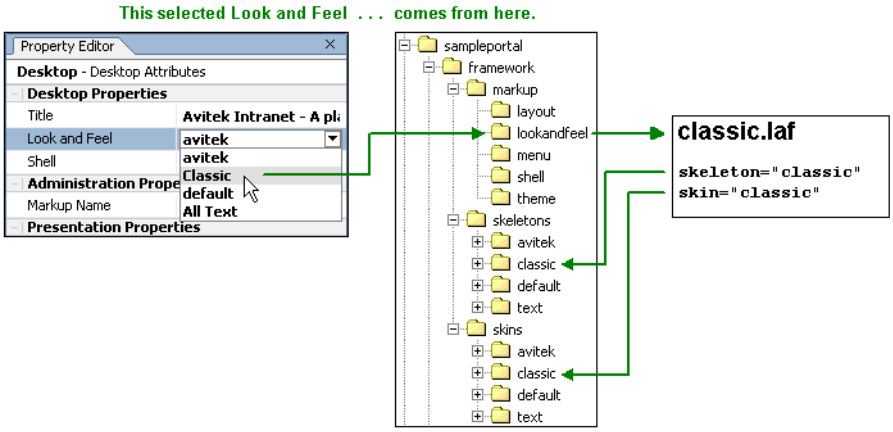
A portal Look and Feel includes two main elements:

Look (skins)
The graphics, cascading style sheets (CSS), and JavaScript files used in a skin.

Feel (skeletons)
The skeleton JSP files that determine the physical boundaries of portal components such as header, footer, book, page, portlet, and portlet title bar.

A portal Look and Feel stems from a single XML file (with a .laf extension). The .laf file determines the name of the Look and Feel and points to the skeleton and skin that the Look and Feel will use, as shown in the following illustration. This simple, extensible framework makes it easy for portal administrators and end users to completely change the appearance of a portal by selecting a different Look and Feel, as shown in Figure A-1.

Figure A-1 Look and Feel architecture



You can apply subsets of skins and skeletons to books, pages, and portlets. These subsets are called themes. Themes let you give individual books, pages, and portlets a different Look and Feel than the rest of the portal desktop. When you select a theme for a book, page, or portlet, the portal framework looks for theme subdirectories under the main skin and skeleton directories used by the selected Look and Feel.

For example, if a desktop Look and Feel uses the /framework/skeletons/default skeleton and you select a theme called "modern" for a portlet, the portal framework looks in the /framework/skeletons/default/modern directory for skeleton JSPs to render just that portlet. Other non-themed portlets are rendered with skeleton JSP files in the /framework/skeletons/default directory.

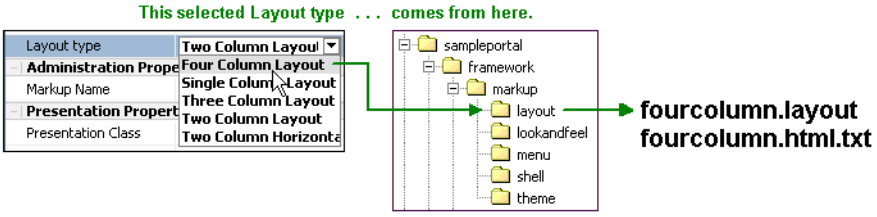
Layouts

Layouts define the placeholders (rows and columns) of a portal page in which portlets can be placed. Layouts also determine whether books and portlets are placed on top of each other (vertical) or beside each other (horizontal) in a placeholder.

A Layout is made up of two files: an XML file (with a .layout extension) and an HTML file (with a .html.txt extension), as shown in the following illustration.

The .layout file is the actual layout structure rendered. The .html.txt file is used to simulate the layout in the WebLogic Workshop Portal Extensions Portal Designer and in the WebLogic Administration Portal.

Figure A-2 Layout architecture



Layouts also have corresponding skeleton JSPs that render them. The desktop's selected Look and Feel determines which skeletons are used. The Creating Skeletons and Skeleton Themes topic contains a table that shows which skeleton JSPs are used.

Navigation Menus

Navigation Menus are used to navigate among books and pages in a portal. WebLogic Portal supplies the following default Navigation Menu styles:

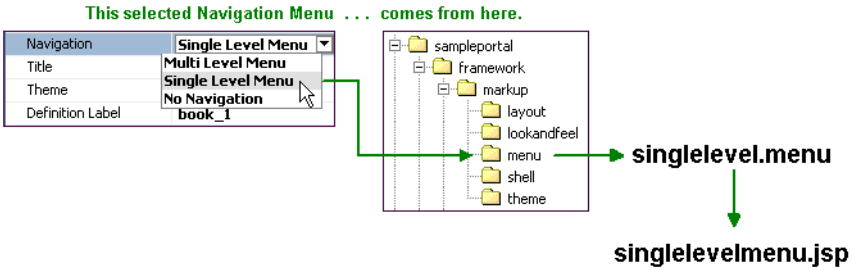
Single Level Menu
: Provides visible layering of page links. Any sub-books and pages appear in rows below the main book navigation.

Multi Level Menu
: Provides a single row of tab-like page links for the top-level pages. Any sub-books and pages appear in a drop-down list for selection. The Multi Level Menu implements JavaScript functionality contained in the skins.

No Navigation
: No navigation is provided.

A Navigation Menu is made up of an XML file (with a .menu extension), as shown in the following illustration. The menu file references a menu class that determines the menu's behavior, and the menu class references the skeleton JSP file that renders the menu.

Figure A-3 Navigation Menu architecture



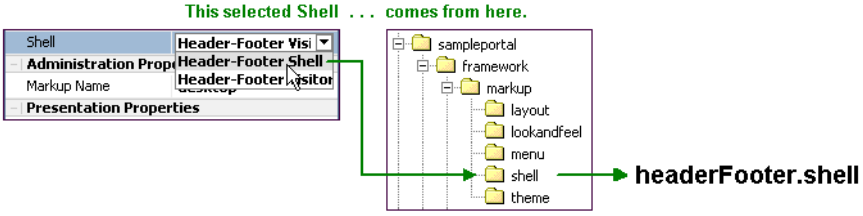
Shells

A shell defines the area around the books and pages of a portal. When a portal is rendered, the shell creates the HTML opening and closing <html>, <head>, and <body> tags and creates the header and footer regions above and below a portal's books and pages.

In the header and footer elements of the shell, you can reference JSPs that display content in the header or footer.

A Shell is made up of an XML file (with a .shell extension), as shown in the following illustration.

Figure A-4 Shell architecture



Shells also have corresponding skeleton JSPs that render them. The Creating Skeletons and Skeleton Themes topic in the WebLogic Workshop help system contains a table that shows which skeleton JSPs are used.

Samples

The Sample applications included with WebLogic Portal 8.1 contain portals that use a default set of Look and Feel resources. Several online help topics provide instructions on opening the portals in WebLogic Workshop Platform Edition to see how the Look and Feel resources are used.

Also, when you add a Portal Web Project to an application in WebLogic Workshop Platform Edition, default Look and Feel resources are included with the project in the `<project>\framework` directory.

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. The following

Portal Component Properties

Table A-1 lists the properties 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 A-1 Properties for All Portal Components

Admininstration Properties	**Markup Name**	Read-only. The unique name of a component markup type. For example, three shell files (markup type "shell") must each use a unique Markup Name inside their .shell files. (Because desktops, books, and pages do not have associated .desktop, .book, and .page files, the exact same Markup Name is used for each.)

Table A-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:
	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:
	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 titlebar, the rendered HTML from the default skeletons will look like this: and the portlet titlebar 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 will be 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 A-2 lists the additional properties that appear when a desktop is selected in the Portal Designer.

Table A-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 A-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 A-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 may 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 may see a different value here.

Table A-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 may 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 may 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 A-4 lists the properties exposed when a book or page is selected in the Portal Designer.

Table A-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 than 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 A-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 an 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 A-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 A-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 A-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 A-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 Thead 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.
	Content URI	Required: Set this to the .JPF file that represents the Java Page Flow.

Table A-6 Portlet Type Properties

Error Page URI	Optional: Designate an error page JSP in this field.

Table A-6 Portlet Type Properties

Portlet Properties	**Title**	Required. Enter the title that will appear in the portlet's titlebar. You can override this title in each instance of the portlet.
	Orientation	Optional. Hint to the skeleton to position the portlet instance titlebar 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" tells the framework that it should attempt to multithread render the portlet. This property can be set to true only if the "Forkable" property is set to "true."
	Forkable	Optional. Lets a portlet developer determine whether or not the portlet is allowed to be multithread rendered. When set to "true," a portal administrator can use the "Fork Render" property to make the portlet multithread rendered.

Table A-6 Portlet Type Properties

Portlet Titlebar	**Icon URI**	Optional. The path (relative to the project) to the graphic to be used in the portlet titlebar. 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 A-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).f

Table A-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 titlebar.
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 A-7 Portlet Instance Properties

Theme	Optional. Select a theme to give the portlet a different look and feel than the rest of the desktop.
Orientation	Optional. Hint to the skeleton to position the portlet titlebar on the top, bottom, left, or right side of the portlet. 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.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

Several framework files have been re-located, replaced or created in the new release, shown in tables

Framework Files

Table A-8 lists commonly edited framework files and their new locations, replacement files, and so on.

Table A-8 Portal Framework File location changes 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	Titlebar 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 A-8 Portal Framework File location changes 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 A-8 Portal Framework File location changes 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 A-9 lists changes to the supported JSP Tags in WebLogic Portal 7.0 SP2.

Table A-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	Personaliation Utilities	es:notNull has been changed to consider an empty string to be “null”. Fomerly, : 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 A-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 A-10 lists changes to the supported JSP Tag libraries in WebLogic Portal Extensions.

Table A-10 JSP Tag Libraries

WebLogic Portal 7.0 SP2	Changes	WebLogic Portal 8.1
ad.tld	none	ad-taglib.jar
cat.tld		cat-taglib.jar

Table A-10 JSP Tag Libraries

WebLogic Portal 7.0 SP2	Changes	WebLogic Portal 8.1
cm.tld	Significant. See Table A-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		Only supported 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 A-11.	
	New in release 8.1	render-taglib.jar: Skeleton rendering tags.
tracking.tld		
um.tld		
util.tld		

Table A-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 A-11 includes specific JSP tags and some changes in the new release.

Table A-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	Resulves to a localized version of the page.

Table A-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	jspContejntUrl	Lets you create URLs to windows and set 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 titlebars, titlebar 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 titlebars.
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 A-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 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.