



BEA WebLogic Portal[®]

Content Management Guide

Version 10.2
Revised: February 2008

Contents

1. Introduction

Introducing Content Management	1-1
Storing Content	1-2
Viewing the Virtual Content Repository	1-3
Using BEA Content Repositories	1-5
Adding Content	1-6
Delivering Content Within Your Portal	1-7
Securing Content	1-8
Content Management in the Portal Life Cycle	1-8
Architecture	1-9
Development	1-9
Staging	1-9
Production	1-10

Part I. Architecture

2. Using Content Repositories

Connecting Repositories to the Virtual Content Repository	2-2
Storing Content in a BEA Default Repository	2-3
Storing Content in a BEA File System Repository	2-3
Storing Content in a Third-Party Repository	2-3
Organizing Your Repository	2-4
Securing your Repository	2-5

3. Configuring BEA Repositories

Working with BEA Repositories	3-1
Working with a Default BEA Repository	3-2
Enabling Library Services for a BEA Repository	3-2
Modifying a BEA Repository	3-3
Disconnecting a Repository	3-8
Working with a BEA File System Repository	3-9
BEA File System Repository Considerations	3-9
Configuring a File System Repository	3-10
Configuring Additional BEA Repositories	3-11
Considerations for Additional BEA Repositories	3-12
Creating Database Objects for the New Repository	3-12
Connecting the New Repository to the Server	3-14
Connecting the New BEA Repository to the Virtual Content Repository	3-18

4. Using Content Folders in Your BEA Repository

Creating a Folder	4-2
Moving a Folder	4-4
Deleting a Folder	4-5
Renaming a Folder or Item	4-5
Changing the Content Workflow for a Folder	4-6

5. Using Content Workflows in Your BEA Repository

Using the Default Content Workflow	5-2
Creating Content Workflows	5-4
Creating or Modifying a Content Workflow Document	5-5
Adding the Content Workflow Document to the Repository	5-17
Assigning Content Workflows to Folders, Content Types, and Content	5-17

Assigning a Content Workflow to a Folder	5-18
Assigning a Content Workflow to a Content Type.	5-18
Assigning a Content Workflow to a Content Item	5-19

6. Using Content Types in Your BEA Repository

Content Types Overview	6-2
Using Content Type Inheritance.	6-3
Using Abstract Content Types	6-3
Using Content Workflows with Content Types	6-4
Working with Content Types.	6-4
Creating a Content Type	6-4
Making a Content Type Abstract	6-6
Deleting a Content Type	6-6
Understanding Content Type Properties	6-7
Supported Data Types.	6-7
Using Nested Content Type Properties.	6-8
Property Options.	6-9
Using Primary Properties	6-10
Setting Property Choice Lists and Default Property Values.	6-11
Using Link Properties.	6-11
Defining Content Properties for Interaction Management	6-11
Define the Properties of a Content Type	6-13
Re-ordering Content Within a Folder Using Properties	6-16
Out-of-the-Box Content Types	6-17

7. Using WebDAV with Your BEA Repository

WebDAV Overview	7-1
WebDAV Guidelines	7-2

Supported Versions of Microsoft Office	7-2
Enabling WebDAV for Repositories.....	7-2
Enabling WebDAV for a non-BEA Repository	7-3
Disabling WebDAV	7-3
Using Content Types with WebDAV	7-3
How WebDAV Determines Which Content Type to Use	7-4
Defining a WebDAV Content Type	7-5
Giving Users Permission to Create Folders.....	7-5
Creating a Content Type That Maps to Microsoft Document Properties.....	7-6
Using WebDAV with Your BEA Repository	7-9
Enabling WebDAV for an Environment.....	7-10
Adding a Microsoft Word Document to a BEA Repository	7-10
Using Windows Explorer to Add a File to the BEA Repository	7-11

8. Connecting to a Third-Party Repository

Working with Third-Party Repositories	8-2
Connecting to a Third-Party Repository	8-2
Logging Into a Third-Party Repository	8-4
Working with a JSR 170-Compatible Repository.....	8-6
Searching within a JSR 170 Repository.....	8-6
Connecting to a JSR 170 Repository	8-8

Part II. Development

9. Delivering Content Within Your Portal

Working with JSP Tags.....	9-1
Retrieving Content with JSP Tags	9-2
Displaying Content with JSP Tags	9-3

10.Adding Content to a BEA Repository

Viewing the Virtual Content Repository	10-2
Working with BEA Repository Content When Using Library Services	10-3
Overview of Library Services	10-4
Adding Content	10-5
Creating HTML Content	10-7
Modifying Content	10-8
Changing the Status of a Single Content Item	10-9
Changing the Workflow Status of Multiple Content Items	10-9
Re-Ordering Content	10-11
Updating Binary Content	10-13
Deleting Content	10-14
Moving Content	10-15
Linking Content	10-15
Renaming Content	10-17
Copying Content	10-17
Previewing Content	10-18
Searching for Content within Your Repository	10-18
Multi-language Searching and Indexing	10-19
Searching for Content By Name	10-19
Searching for Content By Property Value	10-19
Searching the Full Text of Content and Property Values	10-21
Using Versioning	10-22
Checking Out Content	10-22
Checking In Content	10-23
Retiring Content	10-24
Viewing and Searching Version History	10-25

Publishing a Different Version of Content	10-26
Changing the Workflow of Content	10-27

11.Using Display Templates

Using Display Template JSP Tags	11-1
Creating Display Templates	11-2
Creating a wlp-template-config.xml File	11-3
Using the <dt:displaycmtemplate> Within a JSP	11-7
Using the <dt:displaytemplate> Tag Within a JSP	11-11
Customizing the Content Presenter Configuration Wizard	11-11
Using Templates with Content Presenter	11-12
Creating New Display Templates for the Content Presenter Wizard	11-12
Creating a Content Display Template to Reuse	11-12
Creating Display Templates for the Content Presenter Portlet	11-14
Register Your Templates for Content Presenter Portlet	11-15
Using the Content Display Template Wizard	11-21
Updating Content Presenter Display Templates	11-21
Creating a Content Display Template	11-22

12.Using Syndicated Feeds

Using and Modifying the Preconfigured Syndicated Feeds	12-1
URL Format for Syndicated Feeds	12-2
Changing the Search Results Using a URL	12-3
Modifying the Preconfigured Syndicated Feeds	12-4
Creating Custom Syndicated Feeds	12-13
Creating the Syndicated Feed JSPs	12-14
Map the Syndicated Feed JSPs to Display Templates	12-15
Create a Syndication Configuration File	12-15

Securing Syndicated Feeds	12-18
-------------------------------------	-------

Part III. Staging

13.Managing Content Workflows in Your BEA Repository

Understanding Assigned Items	13-1
Viewing Assigned Content Types	13-2
Viewing Assigned Content	13-2
Modifying a Content Workflow	13-3
Deleting a Content Workflow	13-3

Part IV. Production

14.Content Deployment Descriptor

Enabling Library Services in content-config.xml	14-2
Setting User Credentials for an External Repository in content-config.xml	14-2
Editing the content-config.xml File	14-3

15.Caching Content

16.Managing Content Security

Using Delegated Administration for Content	16-1
Setting Delegated Administration	16-2
Setting Visitor Entitlements for Content	16-2

A. Importing Third-Party Content

Preparing to Use BulkLoader	A-2
Creating a Repository	A-2
Creating Appropriate Types	A-2
Preparing a Content Directory	A-2
Preparing Metadata Files	A-3

Configuring and Running BulkLoader	A-7
Editing the BulkLoader Script.	A-7
BulkLoader Command Examples	A-9
BulkLoader Parameter Reference	A-10
Example BulkLoader Script.	A-12

Introduction

Most portals incorporate content into their applications. Content can be anything from advertisements (graphic files), documents (text files), or animation files. Portal content is typically stored in a content repository that is part of a content management system connected to the portal. Developers and administrators then have access to content and can determine how it is viewed by portal visitors.

This chapter includes the following sections:

- [Introducing Content Management](#)
- [Storing Content](#)
- [Adding Content](#)
- [Delivering Content Within Your Portal](#)
- [Securing Content](#)
- [Content Management in the Portal Life Cycle](#)

Introducing Content Management

The content management system in WebLogic Portal allows you to store and access content, track its progress, and incorporate content in your portal applications. It provides an easy integration between creating content and delivering that content to your users. Content contributors use repositories in WebLogic Portal to store and access content. If WebDAV is

enabled, content contributors can also access content through applications such as Microsoft Word. For more information, see [Chapter 7, “Using WebDAV with Your BEA Repository.”](#)

For information about tracking the content retrieved from the virtual content repository, see [Setting Up Events and Behavior Tracking](#) in the *Interaction Management Guide*.

Content repositories are connected to your portal with the Virtual Content Repository. Portal developers use the content API and JSP tools to access the Virtual Content Repository and deliver content to portal visitors. For more information about the Virtual Content Repository, see [“Connecting Repositories to the Virtual Content Repository”](#) on page 2-2.

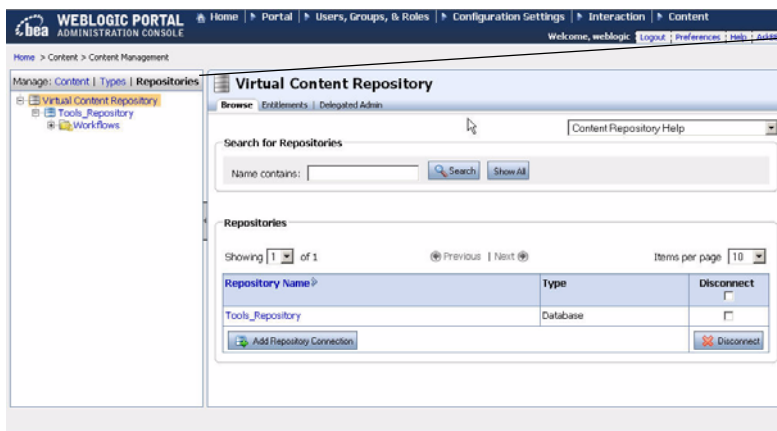
Storing Content

Content repositories are connected to WebLogic Portal using the Virtual Content Repository. When a content repository is connected to the Virtual Content Repository, portal developers and content contributors can access content using WebLogic Portal content tools, including the WebLogic Portal Administration Console, the content API, JSP tags, content selectors, placeholders and WebDAV-enabled applications.

You can also integrate third-party content management systems (including JSR 170-compliant repositories) with WebLogic Portal by connecting them to the Virtual Content Repository. For information about using a third-party repository, see [Chapter 8, “Connecting to a Third-Party Repository.”](#) Once connected to the Virtual Content Repository, content within a third-party repository can be searched and utilized by WebLogic Portal.

Typically, you access the Virtual Content Repository through the WebLogic Portal Administration Console, as shown in [Figure 1-1](#).

Figure 1-1 The Virtual Content Repository within the WebLogic Portal Administration Console



The Virtual Content Repository provides three views: **Content**, **Types**, and **Repositories**.

Viewing the Virtual Content Repository

The Virtual Content Repository provides three views of your content repositories: Content, Types, and Repositories. [Table 1-1](#) lists each view and the tasks you accomplish in each one.

Table 1-1 Summary of the Views of the Virtual Content Repository

Virtual Content Repository View	Associated Tasks
Content view (Click Manage Content)	<ul style="list-style-type: none"> • Add content to the repository • Search for content • Delete content • Modify content <p>If using BEA's library services, you can also:</p> <ul style="list-style-type: none"> • View version history • Check out content • Check in content • For more information about library services, see "Overview of Library Services" on page 10-4.
Types view (Click Manage Types)	<ul style="list-style-type: none"> • Add content types • Modify content types (adding or removing property definitions) • Delete content types
Repository view (Click Manage Repositories)	<ul style="list-style-type: none"> • Connect repositories to the Virtual Content Repository • Edit repository properties such as search settings and caches <p>If using BEA's library services, you can also:</p> <ul style="list-style-type: none"> • Add content workflows • Modify content workflows • Delete content workflows

Using BEA Content Repositories

By default, portal applications are configured to use a single BEA content repository to store your content. BEA repositories use your portal database to store content in reserved tables. However, you can use multiple BEA repositories and configure them to use file system to store your content.

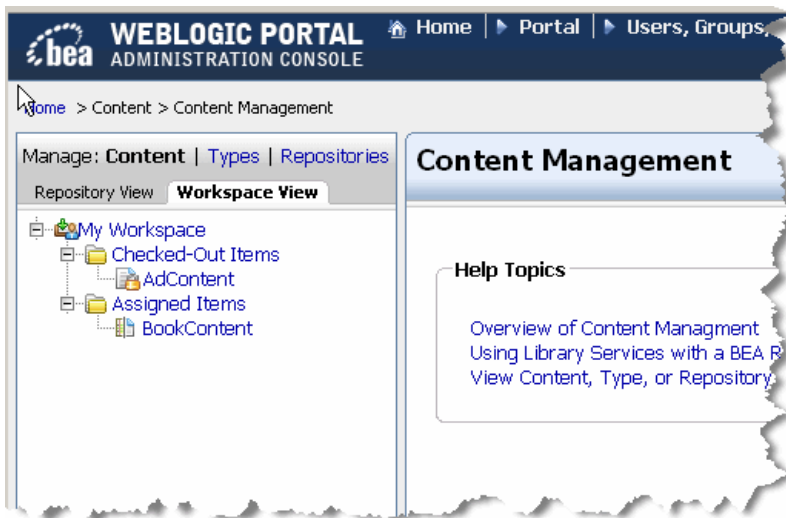
Within a BEA repository, you can:

- Create flexible content types that allow you to link content items, have content items inherit content properties, and define what kind of metadata is associated with content (time/date stamps, author's name, and so on). Content types and their associated properties allow you to search for content within the repository.
- Create a hierarchy of folders to make it easier to categorize your content.
- Preview content, write HTML content on-the-fly, and view past versions of content files.
- Conduct full-text searches within stored content. Full-text search allows you to search for keywords or characters within the content files stored in your repository.

You can also enable BEA repositories to take advantage of BEA's library services. Library services allow users to track versions of content and use content workflows to enforce processes that you want content contributors to use, such as getting approval and retiring outdated content. For more information about library services, see [“Adding Content to a BEA Repository” on page 10-1](#). Library-services content is referred to as versioned content.

If you are using a library services-enabled BEA repository, you can also:

- Create content workflows to route content to appropriate users during the review process.
- Use user-specific workspaces that display which items need a user's approval or show the status of the content the user created as shown in [Figure 1-2](#).
- Use content version control to keep multiple versions of content within your database.

Figure 1-2 Content Workspace View in WebLogic Portal Administration Console

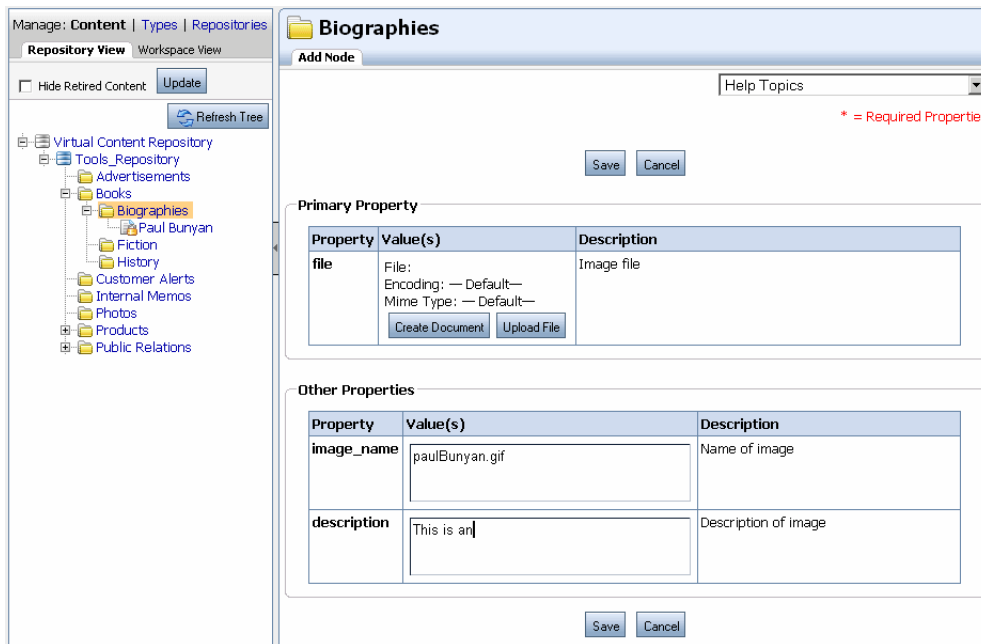
Adding Content

Typically, content contributors add content to a BEA repository through the WebLogic Portal Administration Console, which provides access to your content repositories. Content users can also add content using Windows Explorer, providing the repository is configured to use WebDAV. For more information, see [“Using WebDAV with Your BEA Repository”](#) on page 7-1.

When users add content to a repository, they create metadata for the content file by associating the content with a content type and assigning property values such as date, author, color, and so on. Portal developers use this metadata to retrieve and display content within your portal application. By using content types, portal developers can easily retrieve content from your repository and create content relationships. For example, retrieve all content created by a certain author.

[Figure 1-3](#) shows an example of adding content to a repository.

Figure 1-3 Example of Adding Content to a Repository



For more information about adding content, see [“Adding Content to a BEA Repository”](#) on page 10-1.

Delivering Content Within Your Portal

When a content repository is connected to the Virtual Content Repository, portal developers can deliver that content to your portal users using a variety of WebLogic Portal development tools. These include the content API, JSP tags, and personalization tools. For more information about delivering content in your portal, see [Chapter 9, “Delivering Content Within Your Portal.”](#)

The content management system in WebLogic Portal allows developers to:

- Display content with JSP tags or HTML.
- Programmatically access and display content using the WebLogic Portal API.
- Search across multiple repositories to retrieve content.
- Personalize content delivery by using a wide variety of personalization tools. Personalization tools include content selectors, placeholders, and campaigns, which deliver

dynamic, personalized content to users based on rules and conditions. For more information about personalization, see the [WebLogic Portal Interaction Guide](#).

Securing Content

You can ensure the security of your portal content using Delegated Administration and Visitor Entitlements in the WebLogic Portal Administration Console.

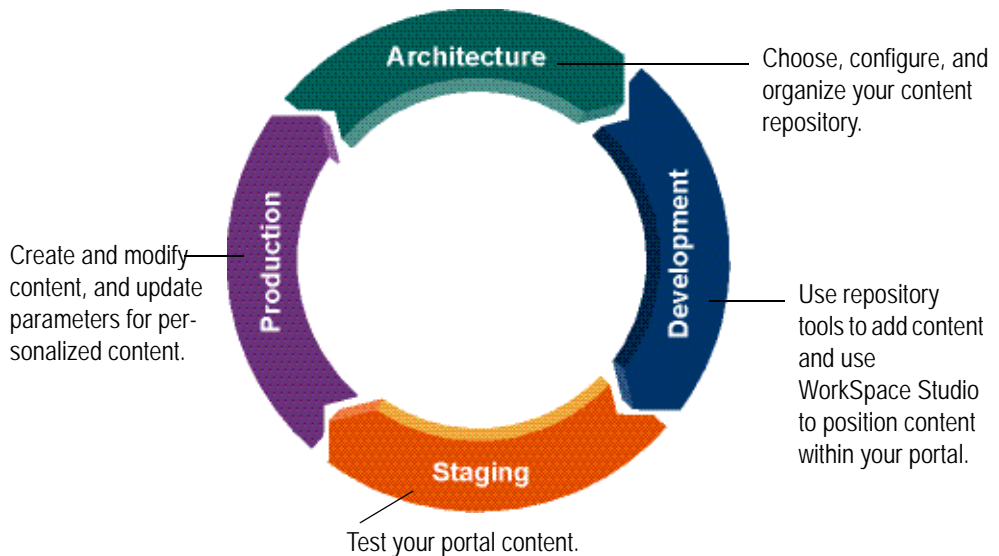
You can use Delegated Administration to determine which users can add or modify content. For example, you can allow only certain users, such as editors, to approve content for publishing, or disallow users from deleting content from your repository. Visitor entitlements allow you to define who can access a portal and what they can do within the portal. For more information about security, see the [WebLogic Portal Security Guide](#).

Content Management in the Portal Life Cycle

The tasks in this guide are organized according to the portal life cycle. For more information about the portal life cycle, see the [WebLogic Portal Overview Guide](#). The portal life cycle contains four phases:

- [Architecture](#)
- [Development](#)
- [Staging](#)
- [Production](#)

[Figure 1-4](#) shows how content management fits into the portal life cycle.

Figure 1-4 How Content Management Fits into the Four Phases of the Life Cycle

Architecture

During the architecture phase, you choose the type of content repository and set it up to match your business needs. This includes creating content types to store content, creating content workflows, and creating content folders to organize your repository. You also plan propagation strategies and determine which content tools you will use.

Development

During the development phase, content contributors add content and developers determine how to present that content within your portal. Content contributors use the WebLogic Portal Administration Console to add and manage content. Using WorkSpace Studio, portal developers use content selectors, placeholders, JSP tags, HTML, and the content API to retrieve and display content.

Staging

The staging phase is when you test your portal and verify that developed content is appearing correctly. You might move iteratively between developing and testing what you created. If you

return to the development phase and make changes, you must redeploy your portal application to see the changes in the staging phase.

Production

This is the phase where you manage your production environment. During the production phase, you use the WebLogic Portal Administration Console to adjust settings, add content, modify content selectors, and create ad campaigns.

Part I Architecture

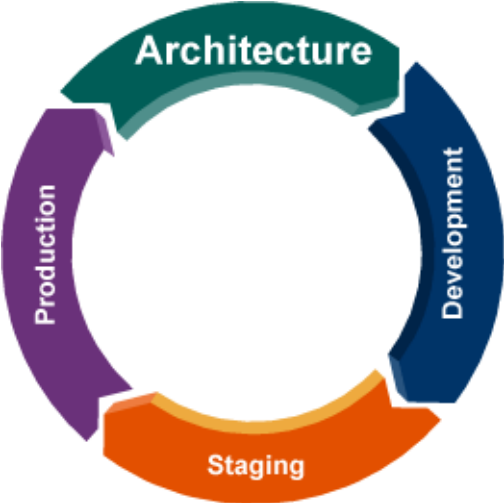
This part contains information to help you complete the architecture phase in the life cycle of content management features for WebLogic Portal.

Part I includes the following chapters:

- [Chapter 2, “Using Content Repositories”](#)
- [Chapter 3, “Configuring BEA Repositories”](#)
- [Chapter 4, “Using Content Folders in Your BEA Repository”](#)
- [Chapter 5, “Using Content Workflows in Your BEA Repository”](#)
- [Chapter 6, “Using Content Types in Your BEA Repository”](#)
- [Chapter 7, “Using WebDAV with Your BEA Repository”](#)
- [Chapter 8, “Connecting to a Third-Party Repository”](#)

After these tasks are done, content contributors can add content to your repository and developers can begin incorporating repository content within your portal application.

For a description of the architecture phase of the portal life cycle, see the [BEA WebLogic Portal Overview Guide](#). The portal life cycle is shown in the following figure:



Using Content Repositories

Before developing content and incorporating it within your portal application, you must choose the types of repositories you will use and connect them to the Virtual Content Repository. This chapter discusses your repository options.

When you create a portal application, it is configured to use a BEA repository. If you decide to use a BEA repository, you need to choose which type of repository to use: the default database repository or a file system repository.

If you already have existing content in another repository, you can connect it to the Virtual Content Repository using an SPI (Service Provider Interface) implementation. For information on how to do this, see the *Content Management SPI Development Guide*.

This chapter includes the following sections:

- [Connecting Repositories to the Virtual Content Repository](#)
- [Storing Content in a BEA Default Repository](#)
- [Storing Content in a BEA File System Repository](#)
- [Storing Content in a Third-Party Repository](#)
- [Organizing Your Repository](#)
- [Securing your Repository](#)

Connecting Repositories to the Virtual Content Repository

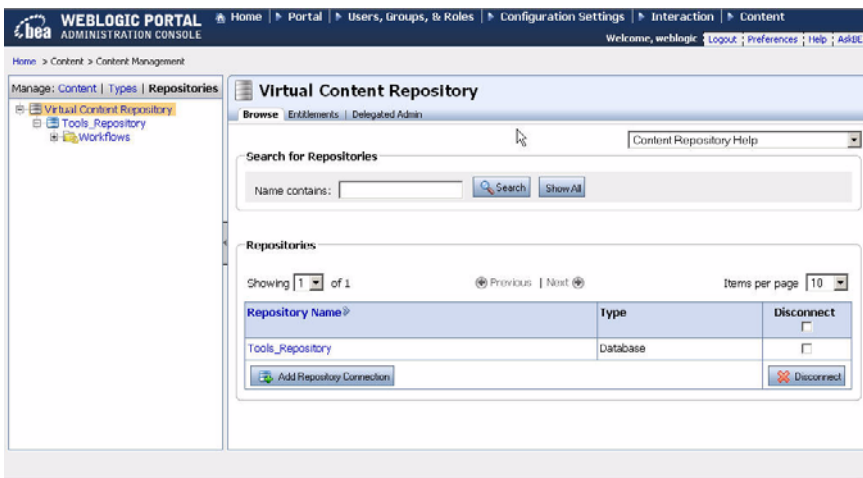
Regardless of what type of repository you use to store your content, all repositories must be connected to the Virtual Content Repository in WebLogic Portal. The Virtual Content Repository allows content contributors to access repositories using the content tools in the WebLogic Portal Administration Console. The Virtual Content Repository also connects your repositories to your portal application so that developers can deliver repository content to your portal users.

Note: You can connect the same repository to multiple portal applications. When using the same repository for multiple applications, only the most immediate application will have access to immediate changes. You can adjust your repository cache settings to ensure that repository views are updated frequently. For more information about setting a repository caches, see “[Modifying a BEA Repository](#)” on page 3-3.

Users (content contributors, administrators, and developers) can access the Virtual Content Repository through the WebLogic Portal Administration Console, shown in [Figure 2-1](#).

Note: If you are using a third-party repository, whether or not you can modify content using the Virtual Content Repository depends on the implementation used to connect to the Virtual Content Repository. For more information about using third-party repositories, see [Chapter 8, “Connecting to a Third-Party Repository.”](#)

Figure 2-1 Viewing Repositories Connected to the Virtual Content Repository



Storing Content in a BEA Default Repository

The default BEA repository comes pre-configured for use within your portal environment. It includes library services which provide content workflows and versioning. For more information about library services, see [“Adding Content to a BEA Repository” on page 10-1](#).

The default configuration uses tables in the portal database to store metadata, content, and content versions. For more information about the BEA default repository, see [“Configuring BEA Repositories” on page 3-1](#).

Storing Content in a BEA File System Repository

BEA file system repositories allow you to use a file system in tandem with the BEA database to store your content. When you use a file system repository, content binary files (stored as binary properties) are stored in the file system you designate, while the metadata associated with the files (content type information) is stored in the a database.

This can increase performance for both data retrieval within your portal and portal tools. Content queries are executed against the database, and results are returned from the file system.

Caution: When using a file system repository, content contributors must use the WebLogic Portal Administration Console to work with content. If content is directly manipulated within the file system, data will be out of sync and behave unpredictably.

When using a file system repository with library services, versioned content is stored in the BEA database. This provides an additional safeguard if the file system is damaged or removed. For more information about library services, see [Chapter 10, “Adding Content to a BEA Repository.”](#)

For more information about using a file system repository, see [Chapter 3, “Configuring BEA Repositories.”](#)

Storing Content in a Third-Party Repository

You can use third-party content repositories with WebLogic Portal. Some third-party content management systems provide connection interfaces for the Virtual Content Repository. BEA provides a set of Java classes called the WebLogic Portal Content Service Provider Interface (SPI), which many content management vendors have implemented. If the vendor of your content management system has implemented the SPI, then adding your repository to the Virtual Content Repository will simply require a configuration within the WebLogic Portal Administration Console.

However, if your vendor has not implemented BEA's SPI, follow the instructions in the [Content Management SPI Development Guide](#).

If you are using a JSR 170-compliant repository, you do not need to write an SPI. WebLogic Portal includes JSR 170 connector to connect to any JSR 170-compliant repository.

When using a third-party repository to store content, you can continue to use that repository's content tools to add and modify content, or depending on the implementation, you may be able to use BEA's content tools.

If the SPI implementation is read only, you need to use the third party repository's tools to create and edit content. However, if the implementation is read/write, then you use either the WebLogic Portal Administration Console or the third-party repository's tools.

Organizing Your Repository

When you use a BEA repository, you need to decide how to logically organize your content to ensure that it is easily retrieved, managed, and used within your portal. Specifically, you should set up a hierarchy of content folders and create the content types that define the content properties. Content properties provide values, such as name and author, for content contributors when they create content.

Tip: Content types and their associated properties provide a flexible way to search for content within the repository. Portal developers can also retrieve content based on its location path within a repository. By thoughtfully planning your content types and content folder hierarchy, you make it easier for portal developers to retrieve and deliver content within your portal application. For complete details on content types, see [Chapter 6, "Using Content Types in Your BEA Repository."](#)

Portal developers can use the metadata (properties) associated with content types to retrieve content files to display within your portal. When you set up content types, you want to be sure to include properties that are useful when retrieving and displaying content, such as the size of an image, whether portal users can click the content, and an expiration date.

Tip: Use metadata to implement a start date so that developers can schedule a go-live time at a future date.

When using a library services-enabled BEA repository, you can also create and assign content workflows to content to ensure that content contributors can route content to the appropriate

approvers before content is published. For complete details on using content workflows, see [Chapter 5, “Using Content Workflows in Your BEA Repository.”](#)

Securing your Repository

When setting up Delegated Administration and Visitor Entitlements, you determine who can add, edit, or delete content within the repository by defining users, groups, and roles.

Visitor Entitlements allow a way for portal administrators to limit what content portal users can view. For example, within a human resources portal, you can entitle content published for managers so that only a user with a manager role can view that content.

You can use Delegated Administration to create administration roles for users and groups who create content or maintain your repository within the WebLogic Portal Administration Console. For example, you can set up a role for users responsible for publishing content, where the role displays only features related to creating and editing content and does not display any other administrative features such as creating types or workflows. You can also create a separate role for creating content types or new folders.

For more information about setting up security, see the [WebLogic Portal Security Guide](#).

For more information about setting up users and groups, see the [WebLogic Portal User Management Guide](#).

Using Content Repositories

Configuring BEA Repositories

Each WebLogic Portal is pre-configured with a default BEA repository. The default repository can be library services-enabled that includes content management features such as a customizable content workflows and versioning.

You can use the pre-configured repository and add additional repositories to suit your needs. If using a third-party repository, see [Chapter 8, “Connecting to a Third-Party Repository.”](#)

This chapter includes the following sections:

- [Working with BEA Repositories](#)
- [Working with a Default BEA Repository](#)
- [Working with a BEA File System Repository](#)
- [Configuring Additional BEA Repositories](#)

Working with BEA Repositories

When working with BEA repositories, you can choose the default database-based repository or switch to a BEA file system repository. Both types allow you to use BEA’s robust library services to manage your content. [Table 3-1](#) lists the features and advantages of both types.

Table 3-1 Repository Types

Repository	Features
BEA Repository (default)	<ul style="list-style-type: none"> • Pre-configured out-of-the-box. • Library services are built-in, including versioning and content workflows. • Stores all content and metadata in the portal database. No additional configuration necessary, unless you want to use multiple repositories.
BEA File System Repository	<ul style="list-style-type: none"> • Stores published content in a file system. • Library services are built-in, including as versioning and content workflows. • Can have higher performance than a default BEA repository. • Can be used with legacy content that is stored in a file system. <p>Note: Does not support transactions as a database-based repository does. If the network connection goes down when adding or modifying content, changes could be lost.</p>

Working with a Default BEA Repository

The default BEA repository comes pre-configured for your portal application.

Before using your BEA repository, ensure that library services are enabled and add any custom properties needed for your environment. For example, add the necessary repository properties to enable integration with full-text search. Repository properties can also indicate if your repository can work in a streaming environment.

Note: Library services allow you to version content and use workflows to manage the content creation process. For more information about library services, see [“Adding Content to a BEA Repository” on page 10-1](#).

Enabling Library Services for a BEA Repository

Library services allow you to version content and use content workflows to route content through an approval and publishing process. If you are using a BEA repository, you should enable library services before organizing your repository.

For more information about using library services, see [Chapter 10, “Adding Content to a BEA Repository.”](#)

Note: Once you have enabled library services for a repository, they cannot be disabled.

To enable library services:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, click the BEA repository for which you want to enable library services.
4. In the **Summary** tab, click **Library Services**.
5. In the **Library Services** dialog, select the **Library Services Enabled** check box and click **Save**.

Modifying a BEA Repository

You can modify the configuration of a BEA repository to suit your environment. For example, you can add custom properties for third-party repositories. You can also configure advanced BEA repository properties, such as cache and search settings.

If you want to use full-text search with your BEA repository, you must add full-text search properties to your repository connection. For more information, see [Integrating Search](#).

To modify a repository connection:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, click the repository you want to modify.
4. In the repository's Summary tab, use the following sections to modify your repository connection:
 - a. In the Repository Details section, you can change the connection class, Datasource JNDI, or make the repository read-only. See [“Repository Details” on page 3-4](#).
 - b. In the Library Services section, you can enable library services.
 - c. In the Properties section, you can add or modify repository properties. See [“Adding Custom Properties” on page 3-5](#).

- d. In the Advanced section, you can modify the repository cache settings. See [“Editing Advanced Repository Properties”](#) on page 3-6.

Repository Details

In the Repository Details section, you can change the connection class, Datasource JNDI, make the repository read-only, or change the password. See [Figure 3-1](#).

- Click **Change Password** to change the password for this repository. Passwords are generally used for third-party repositories.
- Click **Repository Details** to change the connection class, Datasource JNDI, or make this repository read-only. [Table 3-2](#) provides more information about the connection class property.

Table 3-2 Repository Connection Class Property Values

Repository Property	Definition
Repository Class	For BEA repositories, the connection class is the following: <ul style="list-style-type: none">• <code>com.bea.content.spi.internal.ExtendedRepositoryImpl</code> If you are using a file system repository, the connection class is the following: <ul style="list-style-type: none">• <code>com.bea.content.spi.internal.FileSystemRepositoryImpl</code> For third-party repositories, please see your vendor SPI documentation.
Datasource JNDI Name	For the BEA repository, the default value is <code>contentDataSource</code> . JNDI names must be unique to the portal domain.
Read Only	Select to make the repository read only.

Figure 3-1 Edit Repository Details

Adding Custom Properties

You can define properties for repositories within the Virtual Content Repository. For example, if you are using a BEA file system repository, additional properties are required. [Table 3-3](#) lists some examples of repository properties.

Note: A node cannot have more than one binary property in a BEA File System Repository.

To add a property to a repository:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, select the BEA Repository to which you want to add a property.
4. In the Properties section of the Summary tab, click **Add Property**.
5. In the Add Property dialog, enter a name and value for your property.
6. Click **Save**.

A summary of the new repository information is displayed in the Summary tab.

Note: After you disconnect a repository or make any changes to repository properties, WebLogic Portal Administration Console users must log out and log back in to view the changes.

Table 3-3 Examples of BEA Repository Properties

Property	Definition
cm_fileSystem_webpath	When using a file system repository that is available over the web, this property indicates the path of that file system.
cm_fileSystem_webpath	When using a file system repository that is available over the internet, this property indicates the path of that file system.
webdav_enabled	Enables a third-party repository to use WebDAV. WebDav allows users to add content to your repository using Internet Explorer and other WebDAV supported applications. For more information about configuring WebDAV, see Chapter 7, “Using WebDAV with Your BEA Repository.”
WEBDAV_TYPE	Provides the name of the content type to use when adding content to the repository when using WebDAV. For more information about configuring WebDAV, see Chapter 7, “Using WebDAV with Your BEA Repository.”
cm_fireFederatedEvents	Enables events associated with all repository events including content, modifying workflows, and modifying content types. For more information about events, see the WebLogic Portal Interaction Guide .
cm_fireRepositoryEvents	Enables events associated with content changes. Used for full-text search. For more information about events, see the WebLogic Portal Interaction Guide .
CM_DATA_SOURCE	Associates a data source with a repository. Use when you have more than one BEA repository. For more information, see “Configuring Additional BEA Repositories” on page 3-11 .

Editing Advanced Repository Properties

Advanced repository properties include cache settings and enabling different types of search features. [Table 3-4](#) lists the advanced repository properties and how they are used.

Table 3-4 Advanced Repository Properties

Advanced Property	What it does:
Search Enabled	Enables users to search the repository using metadata.
Search Indexing Enabled	Allows content to be indexed for portal search. This enables portal developers to include full-text content search in any portlets that they develop.
Full-Text Search Enabled	Enables users to search the repository using the full-text of the content within the repository.
Streamable	Enables content to be streamed instead of stored in a memory buffer.
Binary Cache	Determines the time-to-live, maximum number of content items that can be cached, and the maximum size on an entry.
Node Cache	Determines the time-to-live and maximum number of content folders that can be cached.

To edit advanced repository properties:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, click the repository you want to modify to view its Summary tab.
4. In the Advanced section, click **Advanced**.
5. In the Edit Advanced Properties for Repository dialog, edit the properties.
6. When finished making changes, click **Save**.

Your modifications display in the Advanced section of the Summary page.

Note: After you disconnect a repository or make any changes to repository properties, WebLogic Portal Administration Console users must log out and log back in to view the changes.

Disconnecting a Repository

You can disconnect any repository within the Virtual Content Repository. When you disconnect a repository, your portal application can no longer access its content.

If you need to delete all content from a repository, you must delete the content store (database or file system). Deleting a datastore or database should only be done by a database administrator.

Note: After you disconnect a repository or make any changes to repository properties, WebLogic Portal Administration Console users must log out and log back in to view the changes.

To disconnect a repository:

1. In the resources tree, select the Virtual Content Repository to see a list of repositories in the Browse tab.
2. In the Browse tab, select the Disconnect check box for the repository you wish to disconnect.
3. Click **Disconnect**.

Note: If a repository was added manually (not through the WebLogic Portal Administration Console), you cannot disconnect it using the console.

Working with a BEA File System Repository

File system repositories allow you to use a file system in tandem with the BEA database to store your content. When you use a file system repository, content binary files are stored in the file system you designate, while the metadata associated with the files (content type information) is stored in the BEA database.

Typically, file system repositories increases performance for data retrieval within your portal. However, not all content management features are compatible with file system repositories. For more information see [“BEA File System Repository Considerations” on page 3-9](#).

BEA File System Repository Considerations

When you use a file system repository, you must organize and manage content according to the same requirements you would have if storing content in a file system. For example, creating a folder in a file system repository creates a folder in the shared directory. For this reason, content requirements exist that help maintain the integrity of the repository and its associated file system.

- In a file system repository, you cannot associate folders with content types. For more information about content folders, see [Chapter 4, “Using Content Folders in Your BEA Repository.”](#)
- A node cannot have more than one binary property in a BEA File System Repository.
- The file system repository will maintain consistency between the name of the content item (node) and the name of the binary property (file):
 - If you create a content item (myNode) without a binary property and later add a binary property (bea.jpg), the content item will be renamed to the name of the binary property (bea.jpg).

- If you create a content item with a name (myNode) and at the same time create its binary property (bea.jpg), the content item will change to the name of the binary property (bea.jpg).
- If you create a content item with the same name and binary property (bea.jpg) and then later update the binary property (diagram.jpg), the content item will be renamed (to diagram.jpg).
- If you create a node with the same name and property (bea.jpg) and then later rename the content item (diagram.jpg), the binary property will be renamed (to diagram.jpg).
- You must always include the file extension when you name or rename content. For example, if you rename a content item called **ball.gif** to be named **ball**, the repository will fail to retrieve the content.

Configuring a File System Repository

When you configure a repository within the Virtual Content Repository, you are creating a connection to the repository's datastore. In the case of a file system repository, the datastore is a file system on your network. When you add a connection file system repository, you also need to add custom properties that direct WebLogic Portal to that file system.

Before You Begin

- The recommended way to implement a file system repository is to modify the properties of the default BEA repository.
- The file system or directory where your file system repository will reside must already exist before creating the repository connection.

Creating a Connection to the New File System Repository

Using the WebLogic Portal Administration Console, edit the connection information for the default BEA repository.

Note: If you want to create an additional repository, see [“Configuring Additional BEA Repositories” on page 3-11](#). After creating the repository, return to this section.

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, click the BEA repository.

4. In the Repository Summary tab, click **Repository Details**.
5. In the **Repository Details** dialog, edit the connection class to be the following:
`com.bea.content.spi.internal.FileSystemRepositoryImpl`
6. Click **Save**.
7. In the **Properties** section, click **Add Property**.
 - a. In the **Name** field, enter `cm_fileSystem_path`.
 - b. In the **Value** field, enter the path to the file system that contains your content. For example:
`/home/myData`.

Note: Be sure that each machine within your portal application cluster has network access to your file system.
 - c. Click **Save**.
8. If your file system is exposed through a web server, you should also set a `cm_fileSystem_webpath` property. This property enables users to access file system content through a web server.
 To add this property, click **Add Property**.
 - a. In the **Name** field, enter `cm_fileSystem_webpath`.
 - b. In the **Value** field, enter the URL of your file system.

Note: For example, if the `cm_fileSystem_path` is set to `/home/myData`, the `cm_fileSystem_webpath` property could reference the same path as `http://mydomain.com/data/myData`.
9. Click **Create**.

Configuring Additional BEA Repositories

You can create multiple content repositories within the Virtual Content Repository to meet your business needs. For example, if you need a physical separation of your content data from your portal application data, then you can create multiple BEA repositories.

Considerations for Additional BEA Repositories

Consider the following when adding repositories:

- While multiple repositories can reside in the same database, each repository must store its data in separate content management tables.
Note: For table separation, in some databases, you create a separate login and schema. In other databases, such as SQL Server and Sybase, table separation is accomplished by creating a separate database.
- To effectively use multiple repositories when using library services, you must use a XA database driver. Note that XA database drivers for PointBase and MySQL database are not supported by WebLogic Portal. In this case use the `LoggingLastResource` global-transactions-protocol in the data source definition instead. For information on how to do this, see [step 7](#).
- For large projects with several thousand content items, you can use separate database instances and minimize changes done in the production environment.
- Each BEA repository must have a unique name. This ensures that each repository can be indexed and searched individually.

Creating Database Objects for the New Repository

In this step, you create database objects for your additional content management database. This involves three basic tasks:

- Using BEA scripts to create the new database or database user for your content management repository, depending on your database vendor.
- Connecting to your new database.
- Running additional BEA scripts to create the content management tables.

To create database objects for the new repository:

1. For Oracle or DB2 databases, create a new database user for your additional content management database. For SQL Server or Sybase, create a new database for your additional content management database objects.

BEA provides sample scripts that can be copied and used to define the database resources that must be configured prior to running any additional `.sql` scripts. For each repository, a separate database or database user must be predefined according to the appropriate sample script. For more details, see the [WebLogic Portal Database Administration Guide](#).

- For **Oracle**, BEA provides the following sample scripts:
`<WebLogic_HOME>/wlserver_10.0/portal/db/oracle/admin/create_tablespace.sql` and `create_users.sql`.
- For **SQL Server**, BEA provides the following script:
`<WebLogic_HOME>/wlserver_10.0/portal/db/sql_server/admin/create_database.sql`.
- For **Sybase**, BEA provides the following script:
`<WebLogic_HOME>/wlserver_10.0/portal/db/sybase/admin/create_devices.sql` and `create_database.sql`.

Note: For both SQL Server and Sybase, the `WEBLOGIC_INDEX` file group must be defined for indexes created using the database-specific `cm_create_tables.sql` and `cm_create_indexes.sql` scripts to execute without errors.

- For **DB2**, BEA provides the following sample scripts:
`<WebLogic_HOME>/wlserver_10.0/portal/db/db2/admin/create_tablespace.sql` and `create_users.sql`.

Note: PointBase is not recommended for a production repository.

2. Connect to the database as the database user created in [Step 1](#).
3. In your domain directory, navigate to the `cmrepo_database.properties` file.
4. Using a text editor, edit the `cmrepo_database.properties` file to match the database that you have created. For more information about this file, see the [WebLogic Portal Database Guide](#).
5. Run the `create_db.cmd/sh` file from your domain directory, using the `-database.properties` parameter to indicate your content management-specific properties file (`cmrepo_database.properties`).

```
create_db.cmd -database.properties=cmrepo_database.properties
```

6. In your domain directory, navigate to the `database.properties` file.
7. Using a text editor, edit the `database.properties` file to match the database that you have created. For more information about this file, see the [WebLogic Portal Database Guide](#).
8. Run the `create_db.cmd/sh` file from your domain directory, using the `-database.properties` parameter to indicate your content management-specific properties file (`database.properties`).

```
create_db.cmd -database.properties=database.properties
```

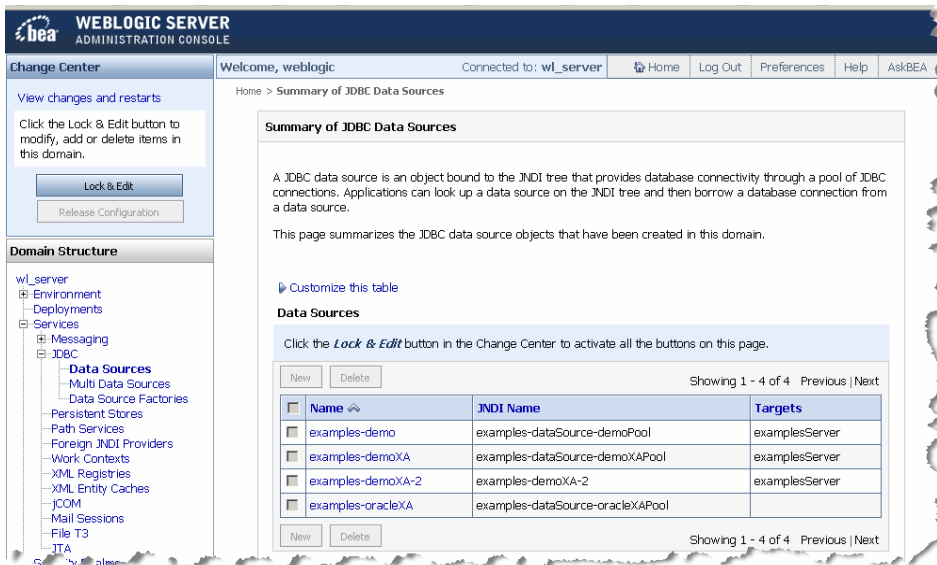
You can now connect your repository to WebLogic Server.

Connecting the New Repository to the Server

Configure your WebLogic Server to point to the new repository by creating a new data source for the repository you want to use. To connect the new repository to the server:

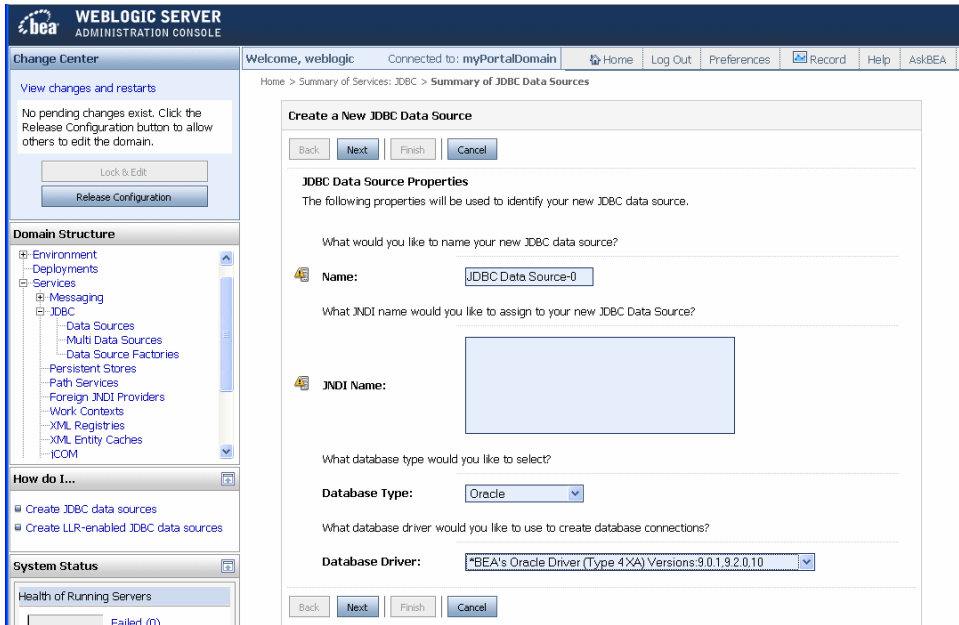
1. Start WebLogic Server for your domain, and log in to the WebLogic Portal Administration Console.
2. In the **Domain Structure** tree, open **Services > JDBC > Data Sources**. [Figure 3-2](#) shows an example of the **Summary of JDBC Data Sources** page.

Figure 3-2 Summary of JDBC Data Sources in the WebLogic Server Console



3. Click **Lock & Edit** to ensure that the server is locked before proceeding.
4. In the **Data Sources** table, click **New**. [Figure 3-3](#) shows an example of the Create a New JDBC Data Source page.

Figure 3-3 Create a New JDBC Data Source—JDBC Data Source Properties Page



5. In the JDBC Data Source Properties page, complete the fields using [Table 3-5](#).

Note: The data source name and JNDI name need to be unique to the domain.

Table 3-5 JDBC Data Source Properties

Field Name	Input
Name	Enter a unique name for your new JDBC Data Source.
JNDI Name	Enter a unique JNDI name for your new data source. This name is used when you configure your repository.
Database Type	Use the drop-down list to select the database type that corresponds with your repository.
Database Driver	Choose a database driver. If using library services, you must use an XA driver.

Note: WebLogic Portal does not support XA database drivers for PointBase or MySQL databases. See [Step 7](#).

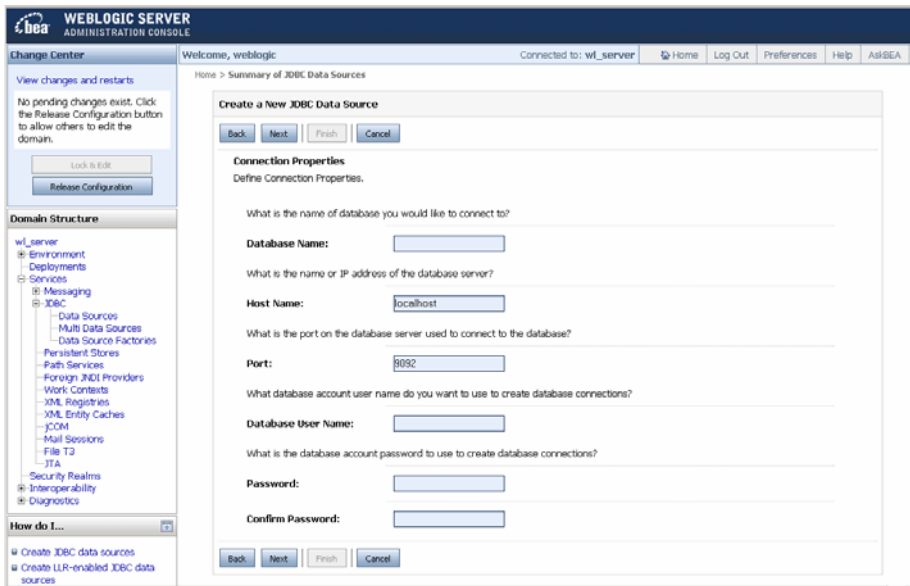
6. Click **Next**.
7. In the Transaction Options page, select any transactions options you require. If you previously selected an XA driver, there are no transaction options to select.

Note: If you use library services, you must select an XA driver.

For PointBase or MySQL databases where an XA Database Driver is not supported or available, select **Supports Global Transactions**, which uses the LoggingLastResource global-transactions-protocol in the data source definition.

8. Click **Next**. [Figure 3-4](#) provides an example of the Connection Properties page.

Figure 3-4 Create a New JDBC Data Source—JDBC Data Source Connection Properties Page



9. In the Connection Properties page, use the information in [Table 3-6](#) to complete the dialog fields.

Table 3-6 Create a New JDBC Data Source—Connection Properties

Connection Property	Description
Database Name	The name of the database you are using.
Host Name	Enter the host name used for the database you are using.
Port	Enter the port number of the port hosting your database.
Database User Name	Enter the database user name for the database login required for this database.
Password	Enter the database password.
Confirm Password	Enter the database password again to confirm.

10. Click **Next**.
11. Optionally, in the **Test Database Configuration** page, click **Test Configuration**. If the database test is successful, click **Finish**.
12. Click **Next**.
13. In the Select Targets page, select one or more targets to deploy your new data source, typically, **AdminServer**.

Figure 3-5 Create a New JDBC Data Source—Select Targets

Create a New JDBC Data Source

Back Next Finish Cancel

Select Targets

You can select one or more targets to deploy your new JDBC data source. If you don't select a target, the data source will be created but not deployed. You will need to deploy the data source at a later time.

Servers
<input checked="" type="checkbox"/> AdminServer

Back Next Finish Cancel

14. Click **Finish**.
15. When finished adding your data source, click **Activate Changes**. The server is updated.

Connecting the New BEA Repository to the Virtual Content Repository

After you have configured the new repository, you need to connect it to the Virtual Content Repository.

1. Use the WebLogic Portal Administration Console to connect to a new repository.
2. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
3. Select **Manage | Repositories**.
4. In the resources tree, select the **Virtual Content Repository**. Figure 3-6 provides an example of the Browse tab within the Repositories section.

Figure 3-6 Browse Tab within the Virtual Content Repository



5. On the Browse tab, click **Add Repository Connection**.

6. In the Add Repository Connection dialog, provide the following information:

Table 3-7 Repository Connection Information

Field	Description
Name	The name you give your new repository. For example: <code>MyNewRepository</code>
Connection Class	<code>com.bea.content.spi.internal.ExtendedRepositoryImpl</code>
Datasource JNDI Name	Use the Datasource JNDI name you created when you connected this repository to your server. The default value of <code>contentDataSource</code> is used for the BEA repository. JNDI names must be unique to the portal domain.
Username	The user name field is only used when connecting to a third-party repository. When configuring a BEA repository, you can leave this blank.
Password	Used only when connecting to a third-party repository. When configuring a BEA repository, you can leave this blank.
Retype Password	Used only when connecting to a third-party repository. When configuring a BEA repository, you can leave this blank.
Enable Library Services	Deselect this check box if you do not want to use library services with this repository.

7. Click **Save**.
8. Within the Repositories section, click the repository you just created to verify that it has been created and view its Repository Summary.

Configuring BEA Repositories

Using Content Folders in Your BEA Repository

Within your BEA repository, you can use folders and subfolders to organize content into logical categories. Additionally, you can associate folders with content workflows and content types, and you can apply security policies to folders.

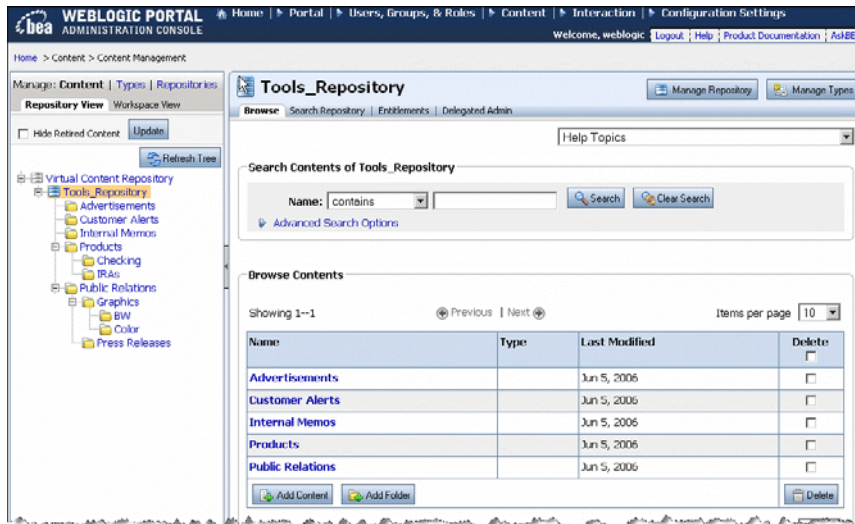
Folders allow you to organize your repository in the following ways:

- Maintain categories for different areas of content, such as human resources, ad campaigns, and video content.
- Manage the content creation process. In a library services-enabled repository, you can associate a folder with a **content workflow**. If a folder is associated with a workflow, all content stored in that folder automatically follows the same process. For more information about content workflows, see [“Using Content Workflows in Your BEA Repository” on page 5-1](#).
- Secure the content files. Using delegated administration, you can add roles to folders that prevent or allow users to access content folders in your repository. Visitor entitlements can also be applied at the folder level. For more information about securing content, see [Chapter 16, “Managing Content Security.”](#)

The BEA repository allows you to use these features at both the folder level and the file level. For example, you can apply security policies on the folders in your repository and maintain workflows and content types at the content level. For more information about content types, see [Chapter 6, “Using Content Types in Your BEA Repository.”](#)

[Figure 4-1](#) shows an example of folders within a repository.

Figure 4-1 An Example Folder Hierarchy within a Repository



This chapter includes the following sections:

- [Creating a Folder](#)
- [Moving a Folder](#)
- [Deleting a Folder](#)
- [Renaming a Folder or Item](#)
- [Changing the Content Workflow for a Folder](#)

Creating a Folder

Use folders to help organize your content. When you create folders, you can also associate folders with content workflows and/or content types. You can only associate a content type with a folder if the repository is not using library services. If library services are enabled, you can associate a folder with a content workflow that is associated with a content type.

When a content workflow is associated with a content folder, all content within that folder follows the workflow. However, if a custom workflow is applied to a content type or explicitly associated with a content item, that workflow overrides the folder workflow. For more

information about using content workflows, see [“How Content Workflows Are Inherited” on page 5-6](#).

When a content type is associated with a folder, you can associate content properties with that folder using that content type. In addition, when users add content to the repository with Internet Explorer or other supported Windows applications, the content they add to the folder is automatically associated with that content type. For more information about adding content to your repository using Windows applications, see, [“Using WebDAV with Your BEA Repository” on page 7-1](#).

Note: You cannot change the content type associated with a folder after you have assigned it. Additionally, after you create a folder, it cannot be associated with a content type.

Note: If you are using a file system repository, you cannot associate a folder with content type.

The following instructions assume your BEA repository is library services-enabled.

To create a folder:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. In the resources tree, select the repository to which you want to add a folder.
4. In the Browse tab, click **Add Folder** to view the Add Folder dialog. [Table 4-2](#) shows the Add Folder dialog.

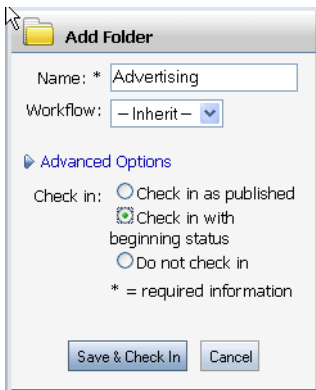
Figure 4-2 Add Folder Dialog

The screenshot shows a standard Windows-style dialog box titled "Add Folder". It features a folder icon in the top-left corner. The main area contains a text input field labeled "Name: *" and a dropdown menu labeled "Workflow: - Inherit -". Below these is a blue link labeled "Advanced Options". At the bottom of the dialog, there is a small text note: "* = required information". Two buttons are positioned at the very bottom: "Save & Check In" and "Cancel".

5. In the Add Folder dialog, type a name for your folder.

6. Optionally, associate a content workflow with the folder by selecting one from the Workflow drop-down list.
7. Optionally, select **Advanced Options** to change the workflow status of the folder itself. By default, new folders are checked into the repository with a published status. By using the Advanced Options, you can choose to have the folder move through the repository workflow, or not to check the folder into the repository. [Figure 4-3](#) shows the available check in options.

Figure 4-3 Using Advanced Options When Creating a New Folder



8. Click **Save & Check In** when finished.

The new folder appears in the repository.

Moving a Folder

After you have created the folders you need, you can re-arrange them within your repository if necessary. Moving a folder also moves all content within that folder.

To move a folder:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. In the resources tree, select the folder you want to move.
4. Right-click the folder you want to move and select **Move** from the context menu.
5. The **Move Content** dialog displays. Click **OK**.

6. Navigate to the repository location to which you want to move the folder (for example, under an existing folder or at the repository root).
7. Right-click the location and select **Paste** from the context menu.
8. The **Paste Content** dialog displays. Click **OK**.

The folder appears in the new repository location.

Deleting a Folder

When you delete a folder, you also delete content within the folder. If you do not want to delete folder content, use the Move command to move content to another location within the repository before deleting the folder, see [“Moving a Folder” on page 4-4](#).

To delete a content folder:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. In the resources tree, right-click the content folder you want to delete.
4. In the Delete dialog, select **Delete**.

The folder no longer appears in the repository.

Renaming a Folder or Item

You can rename content or a content folder in the BEA repository.

To rename a content folder or item:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. In the resources tree, right-click the content folder or item you want to rename.
4. Right-click the content folder or item and select **Rename**.
5. In the Rename dialog, enter in the new name and then click **OK**.

The resources tree now shows the new name for the content.

Changing the Content Workflow for a Folder

You can change the content workflow that a content folder uses. For example, if all of your content uses a default workflow, but you would like a particular content folder (and its contents) to follow a different workflow, you can change the workflow associated with that content.

Note: Content workflows are only available if your repository is library services-enabled.

In some cases, you may not be able to transition to a different workflow, see [“Creating Content Workflows” on page 5-4](#).

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. In the resources tree, click the folder you want to edit.
4. Click the **Summary** tab.
5. If using library services, in the Summary page under Versioning & Workflow, click **Check Out**.
6. Click **Version & Workflow**.
7. In the Update Workflow dialog, select a content workflow and then click **Update**.
The updated workflow information is displayed in the Summary page.
8. Click **Check In**.

Using Content Workflows in Your BEA Repository

If you are using a BEA repository that is library services-enabled, you can enforce a workflow process when users add and publish content in the repository. BEA repositories include one default content workflow. You can create additional content workflows or customize content workflows to suit your business needs.

For example, your content workflow can dictate how extensive the review process should be before content can be published in the portal. You can also customize a content workflow so that when content is published, e-mail is sent to an administrator.

Although you can add new workflows to your repository at any time, if you want to use custom workflows, it is recommended that you create workflows for your repository before you create your content types.

Content workflows are only available in BEA repositories that are library services-enabled. For more information about library services, see [Chapter 10, “Adding Content to a BEA Repository.”](#)

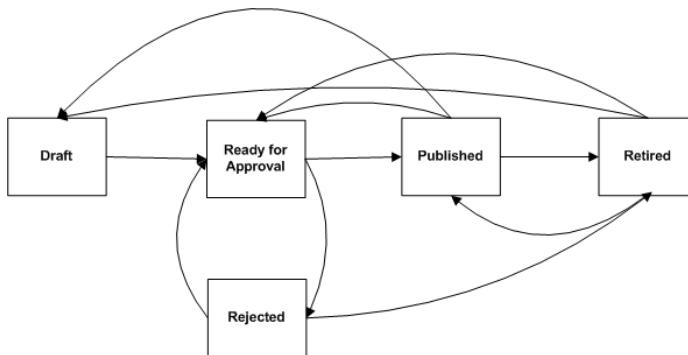
This chapter includes the following sections:

- [Using the Default Content Workflow](#)
- [Creating Content Workflows](#)
- [Adding the Content Workflow Document to the Repository](#)
- [Assigning Content Workflows to Folders, Content Types, and Content](#)

Using the Default Content Workflow

All content created in a library services-enabled BEA repository uses the default content workflow, unless you implement a customized workflow. The default content workflow includes Draft, Ready for Review, Rejected, Published, and Retired states, as shown in [Figure 5-1](#). The default workflow uses delegated administration to prevent or allow users to transition content to different statuses. However, you can visitor entitlements. It is recommended that you use either delegated administration or visitor entitlements, but not both.

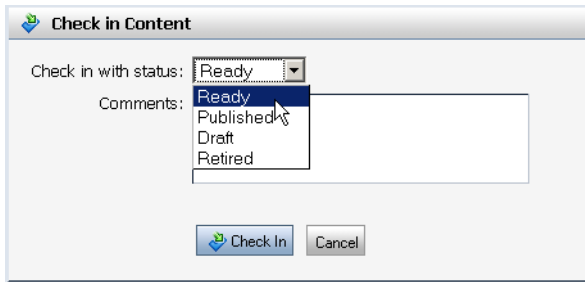
Figure 5-1 Default Content Workflow Diagram



Note: The capabilities you give to a role determines how a user role participates in the content workflow. For example, if a role is not granted the proper capabilities, a user may not be able to transition content to a Published or Retired status.

Users choose from available statuses when they check in content within a library services-enabled repository. [Figure 5-2](#) shows an example of the statuses available within the default content workflow.

Figure 5-2 Changing Workflow Status when Checking In Content in the WebLogic Portal Administration Console



Each transition within the default workflow requires different administrative capabilities. For a complete review of all the available Delegated Administration capabilities for Content Management, see [Setting Delegated Administration on Content Management Resources](#) and [Removing and Editing Delegated Administration on Content Management Resources](#) in the *WebLogic Portal Security Guide*.

[Table 5-1](#) describes the default content workflow statuses and required Visitor Entitlements or Delegated Administration capabilities for transition to different statuses.

Table 5-1 Default Content Workflow

Status	Usage Notes	Required Visitor or Delegated Administration Capabilities
Draft	The initial status for all content items. This status is also used to indicate a content item that is a work in progress and not ready for review. Items in Draft status display in the Assigned Items folder for all users whose role includes Edit and Publish capabilities.	Publish
Ready for Approval	Marks a content item for ready for review and/or publication to the site. Content items that have the status Ready for Approval can be modified only by content administrators who have publish rights.	Publish
Published	Content items that have this status can be accessed by portal content selectors or placeholders according to their content type property values.	Publish

Table 5-1 Default Content Workflow (Continued)

Status	Usage Notes	Required Visitor or Delegated Administration Capabilities
Rejected	The Rejected status is used to re-route a content item to the last known user or group to which it was assigned.	Publish
Retired	The final status of a content item. This status indicates that a content item is no longer in use. Retired content items cannot be retrieved by a content selector or a placeholder.	Publish
Deleted	All versions of the content item are deleted. Only administrators with Delete capabilities can delete content. After deletion content cannot be retrieved by placeholders or content selectors.	Publish

Creating Content Workflows

Content workflows are XML files that store process information. Once these files are added to your repository, you can associate their defined workflow with content. You create and add content workflows by:

- Creating a content workflow document based on the content workflow XML schema. See [Creating or Modifying a Content Workflow Document](#)
- Add the content workflow document to your repository and give it a name. After this document is added to your repository, it is available to associate with content.
- When creating content, content folders, or content types, you can associate the named content workflow with the item.
- When content contributors add content that uses a content type, the content they create must move through the associated workflow, such as Draft > Ready for Review > Published > Retired.
- Using roles in Delegated Administration or Visitor Entitlements, you can allow or prevent users from being able to create and modify content workflows. For detailed information about setting up roles, see [Setting Up Role-Based Authorization](#) in the *Security Guide*.

Creating or Modifying a Content Workflow Document

You can customize your content workflow according to your needs. The following sections contain some examples of customized content workflows:

- [Guidelines for Creating or Modifying a Content Workflow Document](#)
- [How Content Workflows Are Inherited](#)
- [Changing the Workflow to use Visitor Entitlements](#)
- [Changing the Display Names of the Content Workflow States](#)
- [Removing a Status](#)
- [Changing the Default Transitions](#)
- [Assigning Different Capabilities for Different Statuses](#)
- [How to Write an Workflow Action](#)
- [Using the Default Workflow Document to Create a New Workflow](#)

Guidelines for Creating or Modifying a Content Workflow Document

When creating or modifying a content workflow document, keep the following in mind:

Note: For an explanation about how content items transit through a workflow, see [“Understanding Assigned Items” on page 13-1](#).

- **Include an Undefined Status in a Content Workflow** – Include an undefined status in your content workflows to cover situations where statuses do not match, as may happen when migrating workflows. For example, in the default content workflow, content may be designated as Ready for Approval, but in the customized workflow this status does not exist. Instead the new workflow may have two possible statuses: Public and Voting. You can define which status to set when the workflow changes. An undefined transition is indicated by using a value of -1 for the `<from-status>` and `<to-status>` elements as shown in [Listing 5-1](#).
- **Status ID Numbers in a Customized Workflow** – All statuses within the default content workflow are assigned a number: Draft, 1; Ready for Approval, 2; Rejected, 3; Published, 4; and Retired, 5. Use these numbers with their default assignments. This is especially important for publishing content, in which case you must use the integer (`int`) 4. The Workflow Action class requires this number for versioning. Use integers 100 – 999 to add new status ID numbers within a customized workflow.

- **Write Custom Classes to Handle Content** – You can write custom classes to associate with various workflow statuses. For example, you could write a class that sends e-mail to administrators when content is ready for approval. For detailed information on creating Workflow Action classes, see [Content Workflow, Part 1](#) on BEA Dev2Dev. You should put any custom workflow actions in the enterprise application's classpath (`APP-INF/classes`) or in the system classpath. If the workflow actions are shared across enterprise applications, also put them in the system classpath. If your classes do not use anything in content management, you can use only the system classpath.

The best way to write a new class is using an EJB or Utility project in Workspace Studio. Both of these methods will automatically compile and redeploy the classes as needed and you do not have to add the class to enterprise's classpath. You should put any custom workflow actions should in the enterprise application's classpath (`APP-INF/classes`). For information about creating EJB or Utility projects, see [Tutorial: Building Enterprise JavaBeans](#) and [New Utility Project Wizard](#) in Workshop Product Family Documentation.

How Content Workflows Are Inherited

Until you create and associate custom workflows, all content within your repository (folders, content and content types) use the default workflow.

When you associate workflows with content, the following inheritance rules apply:

- Any workflow applied to a folder applies to all content within that folder.
- If a workflow is applied to a content type within the folder, that workflow takes precedence over the workflow applied to the folder.
- If a workflow is explicitly associated with a content item, that workflow takes precedence to any workflow applied to the folder and to any workflow associated with the content type.

Changing the Workflow to use Visitor Entitlements

Note: For content management, you need to use enterprise application-scoped visitor entitlements roles not web application-scoped visitor entitlements roles. For more information about visitor entitlement roles on portal resources, see [Setting Up Role-Based Authorization](#) in the *Security Guide*.

To use visitor entitlements instead of delegated administration, in the default Content Workflow Document, change the value of the <capabilityConstraint> elements as shown in [Table 5-2](#). For example:

- Delegation Administration:
`<capabilityConstraint>can_publish</capabilityConstraint>`
- Visitor Entitlement:
`<capabilityConstraint>can_vis_publish</capabilityConstraint>`

Table 5-2 Delegated Administration versus Visitor Entitlement Values

Delegated Administration	Visitor Entitlements
can_create	can_vis_create
can_view	can_vis_view
can_update	can_vis_update
can_reject	can_vis_reject
can_delete	can_vis_delete
can_associate	can_vis_associate
can_publish	can_vis_publish

Changing the Display Names of the Content Workflow States

If you want change the name of the workflow status (for example, [Figure 5-2](#)), you change the value for the status ID in the default Content Workflow ([Listing 5-1](#)). For example, change

```
<status id="2" text="Ready" /> to <status id="2" text="Submitted" />
```

Tip: The default content workflow is not internationalized. If you want to create a localized content workflow, the XML elements must remain in English. The values of the elements can be in the localized language. For example, `<status id = "3" text="publicado">`.

Removing a Status

To remove a status, you remove the Status ID and any transitions to and from it. For example if you wanted to remove the Retired status from the default Content Workflow ([Listing 5-1](#)):

1. Delete the `<status id>` for the Retired status:

```
<status id="5" text="Retired" />
```

2. Delete the `<to-status>` elements in each of the other statuses that transition to the Retired status. For example, for the transition from Draft to Retired, delete the XML shown in bold:

```
<transition>
  <from-status id="1"/>
  <to-status id="2">
    <action class="com.bea.content.virtual.workflow.ReadyAction"/>
  </to-status>
  <to-status id="4">
    <capabilityConstraint>can_publish</capabilityConstraint>
    <action class="com.bea.content.virtual.workflow.PublishAction"/>
  </to-status>
  <to-status id="5">
    <capabilityConstraint>can_publish</capabilityConstraint>
    <action class="com.bea.content.virtual.workflow.RetireAction"/>
  </to-status>
</transition>
```

3. Delete the Retired transition:

```
<transition>
  <from-status id="5"/>
  <to-status id="1">
    <action class="com.bea.content.virtual.workflow.DraftAction"/>
  </to-status>
  <to-status id="2">
```

```

        <action class="com.bea.content.virtual.workflow.ReadyAction" />
    </to-status>
    <to-status id="4">
        <capabilityConstraint>can_publish</capabilityConstraint>
        <action class="com.bea.content.virtual.workflow.PublishAction" />
    </to-status>
</transition>

```

Tip: Keep the default IDs (1 – 4) for the default statuses. See [“Guidelines for Creating or Modifying a Content Workflow Document”](#) on page 5-5.

Changing the Default Transitions

If you want to change a transition, you must add or remove the transition from the default Workflow Document. [Table 5-3](#) describes the default transitions for each state.

Table 5-3 Default Workflow Transitions

Current Status	Transition to Status
Draft	Ready Publish Retire
Ready	Reject Publish Retire
Reject	Draft Ready
Publish	Draft Retire
Retire	Draft Ready Publish

Assigning Different Capabilities for Different Statuses

The default workflow does not assign different capabilities for different statuses—every status transition requires `can_publish`. This means if you designate an article as Ready for Approval, you can also publish it. For your business needs, you may want to change the user role associated

with transition from a writer to a publisher. For detailed information on assigning different capabilities for different statuses, see [Content Workflow, Part 1](#) on Dev2Dev.

The capability constants are defined in the `com.bea.content.manager.ContentEntitlementHelper` class. For more information, see the WebLogic Portal [Javadoc](#). You'll need the values of these constants if you are creating a custom workflow.

How to Write an Workflow Action

There are two ways you can create an action for a workflow:

- Write a class that implements the `WorkflowAction` interface. For more information, see `com.bea.content.virtual.workflow.WorkflowAction` in the WebLogic Portal [Javadoc](#).
- Extend the `StandardAction` class to your own class. This class defines some methods called `assignNodeToRoles` and `assignNodeToUser` that you can use programmatically to assign nodes to a particular roles, such as writers and publishers. For more information, see `com.bea.content.virtual.workflow.StandardAction` in the WebLogic Portal [Javadoc](#).

Using the Default Workflow Document to Create a New Workflow

The easiest way to create a new content workflow document is to use the default content workflow and save it as a new document.

Tip: Because a workflow affects the entire repository, it is best to play it safe and copy the default workflow to a new workflow, and then apply it to one folder in the repository for testing.

To create a new content workflow document by modifying an existing document:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, click the content workflow you want to copy.
4. In the Workflow File section of the Details tab, click **Download File** and choose to **Save to Disk**. This saves a copy of the content workflow document in the location you specify.

5. Using an XML editor, make your modifications. For more information about the content workflow XML schema, see [Table 5-4](#).

Note: An XML editor reads the workflow schema and the schemas the schema imports. It shows you the elements and attributes that can be added to an XML document.
6. When finished, save the content workflow with a new name.
7. Add the new workflow document to your repository, as described in the next section “[Adding the Content Workflow Document to the Repository](#)” on page 5-17.

[Listing 5-1](#) shows the default workflow document and [Table 5-4](#) provides information about the content workflow XML schema.

Tip: Become familiar with the content workflow schema before you create or modify a content workflow document.

Listing 5-1 Default Content Workflow Document

```
<?xml version="1.0" encoding="UTF-8"?>

<workflow xmlns="http://schema.workflow.virtual.content.bea.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schema.workflow.virtual.content.bea.com">

<transition>
  <from-status id="1"/>
  <to-status id="2">
    <action class="com.bea.content.virtual.workflow.ReadyAction"/>
  </to-status>
  <to-status id="4">
    <capabilityConstraint>can_publish</capabilityConstraint>
    <action class="com.bea.content.virtual.workflow.PublishAction"/>
  </to-status>
  <to-status id="5">
    <capabilityConstraint>can_publish</capabilityConstraint>
    <action class="com.bea.content.virtual.workflow.RetireAction"/>
  </to-status>
</transition>
<transition>
  <from-status id="2">
    <capabilityConstraint>can_publish</capabilityConstraint>
  </from-status>
  <to-status id="3">
    <capabilityConstraint>can_publish</capabilityConstraint>
```

Using Content Workflows in Your BEA Repository

```
<action class="com.bea.content.virtual.workflow.RejectAction"/>
</to-status>
<to-status id="4">
  <capabilityConstraint>can_publish</capabilityConstraint>
  <action class="com.bea.content.virtual.workflow.PublishAction"/>
</to-status>
<to-status id="5">
  <capabilityConstraint>can_publish</capabilityConstraint>
  <action class="com.bea.content.virtual.workflow.RetireAction"/>
</to-status>
</transition>
<transition>
  <from-status id="3"/>
  <to-status id="1">
    <action class="com.bea.content.virtual.workflow.DraftAction"/>
  </to-status>
  <to-status id="2">
    <action class="com.bea.content.virtual.workflow.ReadyAction"/>
  </to-status>
</transition>
<transition>
  <from-status id="4"/>
  <to-status id="1">
    <capabilityConstraint>can_publish</capabilityConstraint>
    <action class="com.bea.content.virtual.workflow.DraftAction"/>
  </to-status>
  <to-status id="5">
    <capabilityConstraint>can_publish</capabilityConstraint>
    <action class="com.bea.content.virtual.workflow.RetireAction"/>
  </to-status>
</transition>
<transition>
  <from-status id="5"/>
  <to-status id="1">
    <action class="com.bea.content.virtual.workflow.DraftAction"/>
  </to-status>
  <to-status id="2">
    <action class="com.bea.content.virtual.workflow.ReadyAction"/>
  </to-status>
  <to-status id="4">
    <capabilityConstraint>can_publish</capabilityConstraint>
    <action class="com.bea.content.virtual.workflow.PublishAction"/>
  </to-status>
</transition>
<!--An undefined transition is indicated by using a value of -1 for the
from-status id. This is used when you change the workflow of a node and
the current status of the node is not a valid from-status in any of the
transitions in this workflow. -->
```

```

<transition>
  <from-status id="-1"/>
  <to-status id="1">
    <action class="com.bea.content.virtual.workflow.DraftAction"/>
  </to-status>
</transition>

<!--The numbers 1-5 are reserved for use within the default content workflow. As
a best practice, use the numbers 100-999 as status ID numbers within a customized
workflow. -->

<beginStatus id="1" />
<status id="1" text="Draft" />
<status id="2" text="Ready" />
<status id="3" text="Rejected" />
<status id="4" text="Published" />
<status id="5" text="Retired" />

</workflow>

```

Table 5-4 lists the XML elements of a content workflow document and the considerations for each element. Listing 5-2 shows the schema itself.

Table 5-4 XML Elements of Workflow Document

Workflow XML Element	Usage
<transition>	Indicates a workflow transition.
<from-status>	<p>Indicates the status from which the transition originates. <from-status> is a child element of the transition element.</p> <p>Note: If you want to include the capability to switch from one workflow to another after the content has been created, you should include an undefined <from-status>. This will allow the workflow to smoothly replace other workflows, if needed. To do this, you should set the <from-status> to -1.</p> <p>The corresponding <to-status> should be a status that is defined in your workflow. For example, you can set up the undefined <from-status> to move to the first status in your workflow. In the default content workflow, the undefined <from-status> transitions to a <to-status> of 1 (Draft).</p>
<to-status>	Indicates the ending status of the transition.

Table 5-4 XML Elements of Workflow Document (Continued)

Workflow XML Element	Usage
<code><action-class></code>	<p>Indicates what action takes place within this transition. The four default action classes are:</p> <ul style="list-style-type: none"> • <code>com.bea.content.virtual.workflow.DraftAction</code> • <code>com.bea.content.virtual.workflow.ReadyAction</code> • <code>com.bea.content.virtual.workflow.RejectAction</code> • <code>com.bea.content.virtual.workflow.PublishAction</code> • <code>com.bea.content.virutal.workflow.RetireAction</code> <p>You can also create your own custom classes. If you use your own custom class, be sure you add them to the portal enterprise application classpath and/or system class path. See “Guidelines for Creating or Modifying a Content Workflow Document” on page 5-5.</p>
<code><capabilityConstraint></code>	<p>Delegated Administration: The syntax for defining a Delegated Administration capability restraint is:</p> <ul style="list-style-type: none"> • Create: <code>can_create</code> • View: <code>can_view</code> • Update: <code>can_update</code> • Delete: <code>can_delete</code> • Reject: <code>can_reject</code> • Associate: <code>can_associate</code> • Publish: <code>can_publish</code> <p>Visitor Entitlements: The syntax for defining a Visitor Entitlement capability is:</p> <ul style="list-style-type: none"> • Create: <code>can_vis_create</code> • View: <code>can_vis_view</code> • Update: <code>can_vis_update</code> • Delete: <code>can_vis_delete</code> • Reject: <code>can_vis_reject</code> • Associate: <code>can_vis_associate</code> • Publish: <code>can_vis_publish</code>
<code><roleConstraint></code>	<p>Indicates what Visitor Entitlement or Delegated Administration role is required for the transition. Be aware that this role is hard-coded in the workflow document and must be manually changed if the role name changes.</p>

Table 5-4 XML Elements of Workflow Document (Continued)

Workflow XML Element	Usage
<beginStatus>	Indicates which status is the beginning status for new content. The number is an int.
<status>	Defines a status ID. The status ID must have both a display name and a number attributed to the status. Numbers 1 – 5 are reserved for the default workflow and cannot be redefined in custom workflows.

Listing 5-2 Workflow Schema (workflow.xsd)

```

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
targetNamespace="http://schema.workflow.virtual.content.bea.com"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://schema.workflow.virtual.content.bea.com"
elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:element name="workflow">
    <xs:annotation>
      <xs:documentation>The content workflow</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="transition" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="from-status">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="capabilityConstraint"
                      type="xs:string" maxOccurs="unbounded" />
                    <xs:element name="roleConstraint"
                      type="xs:string" maxOccurs="unbounded" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:attribute name="id" type="xs:integer"
                use="required"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

        </xs:complexType>
    </xs:element>
    <xs:element name="to-status" maxOccurs="unbounded">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="action"
                    maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:attribute name="class"
                            type="xs:string" use="required" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="capabilityConstraint"
                    type="xs:string" maxOccurs="unbounded" />
                <xs:element name="roleConstraint"
                    type="xs:string" maxOccurs="unbounded" />
            </xs:sequence>
            <xs:attribute name="id" type="xs:integer"
                use="required" />
        </xs:complexType>
    </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="status" maxOccurs="unbounded">
    <xs:complexType>
        <xs:attribute name="id" type="xs:integer"
            use="required" />
        <xs:attribute name="text" type="xs:string"
            use="required" />
    </xs:complexType>
</xs:element>
<xs:element name="beginStatus" maxOccurs="1">
    <xs:complexType>
        <xs:attribute name="id" type="xs:integer"
            use="required" />
    </xs:complexType>
</xs:element>

```

```
        </xs:sequence>
    </xs:complexType>
</xs:element>

</xs:schema>
```

Adding the Content Workflow Document to the Repository

Content workflows must reside in your repository before you can associate them with content. You do this using the Content Management section of the WebLogic Portal Administration Console.

To create a new content workflow document by modifying an existing document:

To add a content workflow to your repository:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, select the repository to which you want to add a workflow.
4. In the Summary tab, click **Workflows**.
5. In the Browse Workflows section, click **Add Workflow**.
6. In the Add Workflow dialog, enter the name, description, and the XML file for the workflow.
7. Click **Save**.

Assigning Content Workflows to Folders, Content Types, and Content

Workflows can be associated with content types or folders. Workflows define the process that content contributors must follow when adding content to a repository.

Note: If you intend to allow users to change the associated workflow for content, you must include an undefined status within the workflow document, see [“Guidelines for Creating or Modifying a Content Workflow Document” on page 5-5](#).

When a content workflow is associated with a content folder, all content within that folder follows the workflow. If content within the folder uses a content type that is also associated with a content workflow, the content type workflow overrides the folder workflow. You can also change the content workflow for an individual content item. If the content workflow is set at the content item-level, that content workflow overrides any other associated content workflow.

Assigning a Content Workflow to a Folder

After you have added a content workflow to your repository, you can associate it with a content folder. You can choose to associate content workflows at the folder level, or only with content types.

To associate a content workflow with a folder within the Virtual Content Repository:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. In the resources tree, click the folder to which you want to associate a content workflow.
4. Click the Summary tab.
5. In the Summary tab under Versioning & Workflow, click **Check Out**.
6. Click **Versioning & Workflow**.
7. In the Update Workflow dialog, select a content workflow from the drop-down list, and then click **Update**.
8. Click **Check In**.

You can also use the Content Management APIs to assign a content workflow to a folder:

```
com.bea.content.federated.IWorkflowManager.setNodeWorkflow  
(ContentContext context, ID nodeId, ID workflowId)
```

For information about the API, see the WebLogic Portal [Javadoc](#).

Assigning a Content Workflow to a Content Type

When you create a content type, you can assign a content workflow to a content type. For instructions on how create a content type, see [“Understanding Content Type Properties” on page 6-7](#).

When a content workflow is associated with a content type, all content of that type uses that workflow. However, users can change the workflow for a particular content item, as described in the next section, [“Assigning a Content Workflow to a Content Item” on page 5-19](#).

If you assign a content workflow to a content type other than the WebLogic Portal default workflow, you can use the **View Assigned Workflows** tab to view which content types use particular workflows. For more information, see [Chapter 13, “Managing Content Workflows in Your BEA Repository.”](#)

You can also use the Content Management APIs to assign a content workflow to a content type:

```
com.bea.content.federated.IWorkflowManager.setTypeWorkflow
    (ContentContext context, ID typeId, ID workflowId)
```

For information about the API, see the WebLogic Portal [Javadoc](#).

Assigning a Content Workflow to a Content Item

When content is added to a folder, it is automatically associated with the content workflow associated with that folder or to the content type it uses. For more information, see [“How Content Workflows Are Inherited” on page 5-6](#). However you can explicitly assign content a content workflow.

If you assign content to a different workflow than the default WebLogic Portal workflow, you can use the **View Assigned Workflows** tab to view which content types use which particular workflows. For more information about managing workflows, see [Chapter 13, “Managing Content Workflows in Your BEA Repository.”](#)

To assign a content workflow to a content item:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select the **Workspace View**.
4. In the Assigned Items folder, click the content you want to edit.
5. In the Versioning & Workflow section of the Summary tab, click **Check Out**.
6. In the Checked-Out Items folder, select the content you want to edit.
7. In the Summary tab, click **Versioning & Workflow**.

8. In the Update Workflow dialog, select a content workflow to use and click **Update**.
9. If finished modifying your content, click **Check In**.

You can also use the Content Management APIs to assign a content workflow to a content item:

```
com.bea.content.federated.IWorkflowManager.setNodeWorkflow  
    (ContentContext context, ID nodeId, ID workflowId)
```

For information about the API, see the [WebLogic Portal Javadoc](#).

Using Content Types in Your BEA Repository

Content types are used to define the metadata that you can associate with content. When content contributors add content to your content repository, they can associate the content with a content type.

For example, when adding an image file to the repository, a content contributor may choose to associate the image file with the “image” content type. The image content-type may include properties such as width and height, color or gray scale, and alternative text.

You can use content types and their associated properties to search for content within the repository. By thoughtfully planning your content types, you make it easier for portal developers to retrieve and deliver content within your portal application.

The BEA repository includes some predefined content types; you can create your own types to meet your business needs. As a best practice, design and add most of your content types before releasing the content repository to content contributors to add content. Although you can add or update content types at any time during your development process, you cannot delete a content type after it has been associated with content.

If your BEA repository uses library services, you can associated content types with content workflows. Content workflows enforce an approval and publication process for content. When you associate a content workflow with a content type, all content of that content type follows the same publication process, unless the content item has been assigned its own workflow. For example, the BEA default workflow includes the following process: Draft > Ready for Review > Published > Retired. For more information about content workflows, see [Chapter 5, “Using Content Workflows in Your BEA Repository.”](#)

This chapter includes the following sections:

- [Content Types Overview](#)
- [Working with Content Types](#)
- [Understanding Content Type Properties](#)
- [Out-of-the-Box Content Types](#)

Content Types Overview

It's recommended that you create most of your content types before any content is added to your repository. However, you can add new content types at any time. There are five important aspects of content types:

- **Property definitions.** Property definitions define what type of information can be associated with content. You define the type of information to enter, any default values, whether a property is required, and so on. For more information about content type properties, see [“Understanding Content Type Properties” on page 6-7](#).
- **Abstract Content types.** You can create an abstract content type that can be used as a building block within other content types, but not be directly associated with content. You also create non-abstract content types that have no such restriction. For more information, see [“Using Abstract Content Types” on page 6-3](#).
- **Content type inheritance.** You can use content type inheritance to add properties to different content types. For example, you can create a content type that inherits some of its property definitions from another content type. For more information about type inheritance, see [“Using Content Type Inheritance” on page 6-3](#).
- **Content workflow.** You can associate a content types with a content workflows, which means that when you add content of a certain type, the associated workflow is initiated. By default, content types use the default workflow. For more information about content workflows, see [“Using Content Workflows in Your BEA Repository” on page 5-1](#).
Note: You must be using library services to use content workflows. For more information about library services, see [“Working with BEA Repository Content When Using Library Services” on page 10-3](#).
- **Nested Content Type Properties.** You can create a property that nests a content type within the content type you are creating. When you nest a content type as a property, the properties of that nested type are viewed as multiple values for that property. For more

information about nested content type properties, see [“Using Nested Content Type Properties” on page 6-8](#).

Using Content Type Inheritance

When you create a content type, you can indicate whether it inherits properties from a parent content type. When you indicate that a content type inherits properties from another type, all properties from the parent type are added to the child type. This can be useful when creating content types that require the same subset of properties. For information on creating a content type that inherits properties from another type, see [“Working with Content Types” on page 6-4](#).

Tip: To associate a content type with a folder, you must have that content type inherit from the pre-defined CM_FOLDER_BADGE content type. For more information about folders, see [Chapter 4, “Using Content Folders in Your BEA Repository.”](#)

For example, you can create a content type for all product content in your repository. This PRODUCT content type includes properties that relate to most products such as such as SKU, price, and manufacturer. Then, when you create a content type for a specific product, such as a Computer content type, you can indicate that the computer content type inherits property definitions from a parent content type called Product. This adds all of the properties from the Product content type (SKU, price, and manufacturer) to the Computer content type (the child content type). In addition to the properties inherited from the Product content type, you can add computer-specific properties to the Computer content type such as Memory, Hard Disk Space, and Processor.

With content type inheritance, each time you edit a parent content type, the children content type is also modified. Using the previous example, if you added a new property to the PRODUCT content type, that property would automatically be added to the COMPUTER content type.

Using Abstract Content Types

Abstract content types allow you to create re-usable property sets to use as building blocks in other content types. An abstract content type can only be used is within another content type, by using a nested content type or with inheritance. If you mark a content type as “abstract”, you prevent content contributors from creating content based on that content type. Use abstract content types to ensure consistency across other content types.

For example, if you want to include address information in multiple content types, you can create an abstract content type called “address” and then add that content type to another content type.

For more information about nested content types, see [“Using Nested Content Type Properties” on page 6-8.](#)

For more information about creating abstract content types, see [“Making a Content Type Abstract” on page 6-6.](#)

Using Content Workflows with Content Types

When you create a content type, you can associate a content workflow with that type. This means that all content of that content type will be automatically associated with that workflow. For example, if you have a content type specific to public relations content, you can associate that content type with the custom workflow that you designed for public relations content. For more information about creating custom workflows, see [Chapter 5, “Using Content Workflows in Your BEA Repository.”](#)

Note: Content workflows are only available if your BEA repository uses library services. For more information about library services, see [“Overview of Library Services” on page 10-4.](#)

Working with Content Types

Content types are stored and managed within the Virtual Content Repository, which is accessed with the WebLogic Portal Administration Console. You can add, modify, and delete content types.

Note: You can allow or prevent users from being able to create and modify content types using Delegated Administration. For detailed information about setting up Delegated Administration on content resources, see the [WebLogic Portal Security Guide.](#)

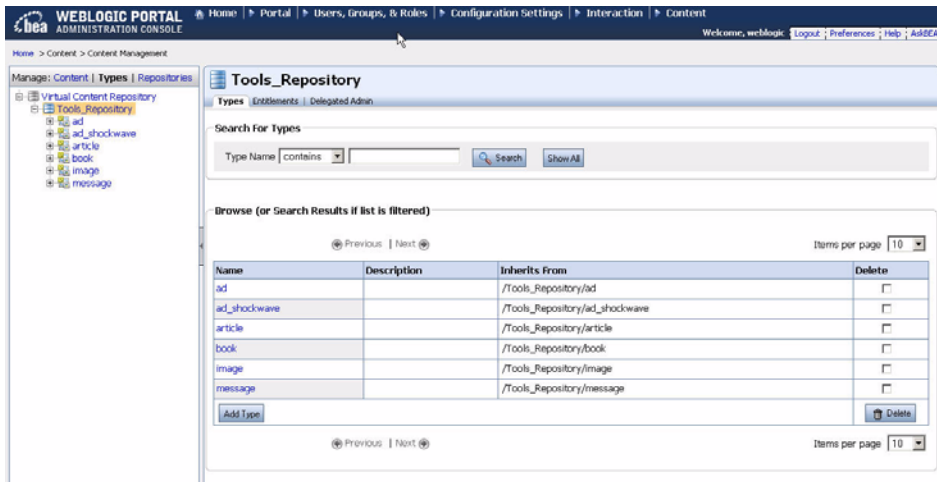
Creating a Content Type

You create content types within the WebLogic Portal Administration Console.

To create a new content type:

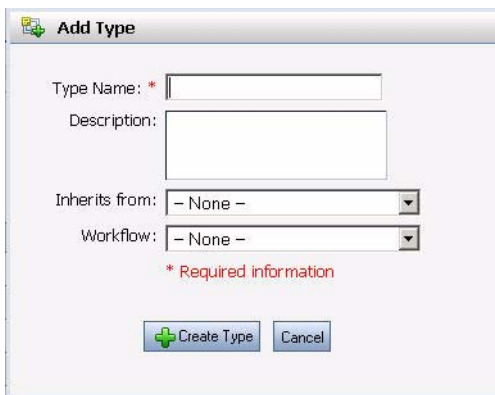
1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management.**
2. Select **Manage | Types.**
3. In the resources tree, select the repository in which you want to create a type. [Figure 6-1](#) shows an example window.

Figure 6-1 Types Tab within the Manage | Types View



4. Within Browse Types, click **Add Type**.
5. In the Add Type dialog, enter a name and description for your content type.

Figure 6-2 Add Type Dialog



6. Optionally, select a content type from which to inherit additional content properties. For more information, see [“Using Content Type Inheritance”](#) on page 6-3.

Note: If you want to associate a content type with a folder, you must create a content type that inherits from the CM_FOLDER_BADGE content type.

7. Optionally, select a workflow to use for this content type.

Default is the name of the default workflow that comes with your BEA repository. If other workflows have been created for this repository, they display in the drop-down list in addition. For more information about workflows, see [“Using Content Workflows in Your BEA Repository” on page 5-1](#).

8. When finished, click **Create Type**.

The Browse tab displays the new type.

Making a Content Type Abstract

You can make a content type abstract. For more information about abstract content types, see [“Using Abstract Content Types” on page 6-3](#).

When you create an abstract content type, users cannot associate content with that content type unless it is inherited or used as a nested property. For more information about inheritance and nested content types, see [“Working with Content Types” on page 6-4](#).

To make a content type as abstract:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Types**.
3. In the resources tree, select the repository in which your content type resides.
4. Click the content type you want to make abstract.
5. In the Summary tab, click **Name & Status**.
6. In the Edit dialog, select the **Is Abstract** check box.
7. Click **Save**.

Deleting a Content Type

You can delete a content type from your repository under certain circumstances. Specifically, you cannot delete a content type under the following conditions:

- The content type has been associated with content.
- The content type has been inherited by another content type.

- The content type is used as a nested content type property.

To delete a content type:

1. In the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Click **Manage | Types**.
3. In the resources tree, right-click the content type and select **Delete** from the pop-up menu.
4. A pop-up window displays asking you to confirm that you want to delete the content type. If you want to proceed, click **Delete**.
5. In the Delete confirmation dialog, click **Delete**.

Understanding Content Type Properties

When creating content types, it is important to understand the properties you can use to define content. The property types you use determine how content is used in your portal. The more properties that are associated with a piece of content, the more granular your searches can be when retrieving content associated with that type.

You can also use content properties in conjunction with interaction management tools, such as content selectors and campaigns, to define when content expires and stops displaying to your users. For more information, see [“Defining Content Properties for Interaction Management” on page 6-11](#).

Note: For more information about interaction management, see the *WebLogic Portal Interaction Management Guide*.

Property definitions for content type properties include multiple attributes, such as data types, primary properties, default values, and read-only and searchable options.

Supported Data Types

Properties can be of different data types. Each data type represents a unique definition that helps you capture specific information about content. For example, to associate content files with content, you must define a property that uses the binary data type. Users can then associate binary files (documents, graphics, and so on) with content. You can also define properties as string data types, which allows users to associate a description of the file with the content.

[Table 6-1](#) lists the data types that can be used.

Table 6-1 Supported Data Types

Data Type	Value Definition
Boolean	A data type that can have one of two values: true or false.
Long Integer	A 64-bit integer
Number with Decimal (Double)	A 64-bit integer that includes one decimal.
String	A variable length character string. The length of this string is dependent on limitations of the database you are using.
Date/Time	A data type that contains the date and time. When this definition is used, users can define the value using a calendar and time entry control.
Binary	A data type that contains a binary file.
Nested Content Type	A data type that contains another content type. Nested Content Type properties cannot be marked as primary. For more information about nested content types, see “Using Nested Content Type Properties” on page 6-8.
Link	A data type that contains a link to another content item within the Virtual Content Repository.

Using Nested Content Type Properties

You can create a property that nests a content type within the content type you are creating. When nesting a content type as a property, the properties of that nested type are grouped together to make up the value of the nested property. For example, if you create content types that require an address to be entered, you can create an address content type and then nest that content type within each content type where an address needs to be entered. [Figure 6-3](#) shows how a nested content type displays when filling in content properties.

For details on adding a nested type property, see [“Define the Properties of a Content Type”](#) on page 6-13.

Figure 6-3 Example of a Nested Content Type Property

publisher	<input type="text"/>	Publisher of the book
state	<input type="text"/>	State book was published
Author's Address	Street:	<input type="text"/>
	City:	<input type="text"/>
	Zip/Postal Code:	<input type="text"/>
	Phone Number *:	<input type="text"/>

Property Options

When you define a property, you define how that property definition behaves within the context of your content type. For example, you can make a property required or read-only. [Table 6-2](#) lists the property options that are available.

Table 6-2 Available Property Options

Property Option	Usage
Required	Makes this property mandatory when creating content.
Read Only	Makes this property read-only. If a property is marked as read-only, its value cannot be edited.
Primary Property	Makes this property the primary property for the content type. Use primary properties within content searches and placeholders to optimize queries. For example, you can retrieve a list of content according to the primary property of its content type. A content type can have only one primary property.

Table 6-2 Available Property Options (Continued)

Property Option	Usage
Searchable	Makes this property accessible through developer content queries. Sometimes, you will not want properties to be searchable. For example, salary data might be confidential and not searchable.
Is Explicit	<p>Map this property value directly to a value contained in the content management database where your content is stored.</p> <p>When you enable Is Explicit, you must indicate the database column within the CM_NODE database table to which you want to derive the property value. For more information about the content management tables, see the WebLogic Portal Database Guide.</p> <p>Note: When a database schema modification occurs as part of adding an explicit property definition that requires modifying the CM_NODE table, you must restart the WebLogic Portal server(s).</p>

Using Primary Properties

Portal developers use primary properties when retrieving content to display to portal users. To display binary content in your content selectors and placeholders, you must assign binary properties as the **Primary Property**.

Primary Properties with Nested Content Types or Content Type Inheritance

When you create a content type that inherits its properties from another content type, if the inherited content type contains a primary property, it will be unmarked as primary. See [Figure 6-4](#).

Figure 6-4 Unmarked Primary Property



It is not possible to have a content type inherit a primary property or to include a primary property within a nested content type.

Setting Property Choice Lists and Default Property Values

You can set default values for the properties you define within your content types. Default values ensure consistent data entry when content contributors add content to your repository. You can also provide choice lists and limit the entries to the values included in the choice list. For information on how to do so, see [“Define the Properties of a Content Type” on page 6-13](#).

Using Link Properties

You can create properties that allow content contributors to associate content items. Content contributors can link to content within the same or different repositories within the Virtual Content Repository. For example, if you have related content items that are stored in different folders, you can use content link properties to create relationships among content items. These relationships are used by developers in their content queries when retrieving content to display in your portal.

Link properties can also be multi-valued to allow content contributors to link to multiple content items. For detailed instructions on adding a link property, see [“Define the Properties of a Content Type” on page 6-13](#).

Defining Content Properties for Interaction Management

Targeting users with content is one of the most important aspects of including content in your portal. You can do this by using interaction management tools such as placeholders, content selectors, campaigns, and JSP tags. For more information about using these tools, see the [WebLogic Portal Interaction Management Guide](#).

For example, each time an employee visits your intranet portal, you can display a different picture from the company picnic. Another example is running a campaign asking employees to renew their benefits in the internal Human Resources portal from November 1 to 30. During this period the campaign displays an Open Enrollment graphic in the portal header.

Adding specific properties to your content types can ensure that developers can take full advantage of the available interaction management tools. For example, if you want to use goal setting to end campaigns based on the number of times content is clicked, you must use the `adTargetUrl`, `adTargetContent`, or `adMapName` properties.

Tip: Use an abstract content type to add interaction management properties to multiple content types within your repository. To do this, create a content type that includes each interaction management property and then make the type abstract. You can then add this abstract type to any content type within your repository. See [“Using Abstract Content Types” on page 6-3](#).

Table 6-3 describes the available properties for content types.

Table 6-3 Interaction Management Properties for Content

Content Property	Type	Description
adTargetUrl	String	<p>Makes an image clickable and provides a target for the clickthrough, expressed as a URL. The Event Service records the clickthrough.</p> <p>Note: Depending on how you want to identify the destination of the ad clickthrough, use adTargetUrl, adTargetContent, or adMapName.</p>
adTargetContent	String	<p>Makes an image clickable and provides a target for the clickthrough, expressed as the content management system’s content ID. The Event Service records the clickthrough.</p> <p>To view a content item’s unique ID, select the content item in the WebLogic Portal Administration Console and view the description in the Edit Content window.</p> <p>Note: Depending on how you want to identify the destination of the ad clickthrough, use adTargetUrl, adTargetContent, or adMapName.</p>
adMapName	String	<p>Makes an image clickable using an image map; you can specify one or more targets. The value for this attribute is used in two locations:</p> <ul style="list-style-type: none"> In the anchor tag that makes the image clickable: <code> </code> In the map definition: <code><map name=value></code> <p>If you specify a value for adMapName, you must also specify a value for adMap.</p> <p>Note: Depending on how you want to identify the destination of the ad clickthrough, use adTargetUrl, adTargetContent, or adMapName.</p>

Table 6-3 Interaction Management Properties for Content (Continued)

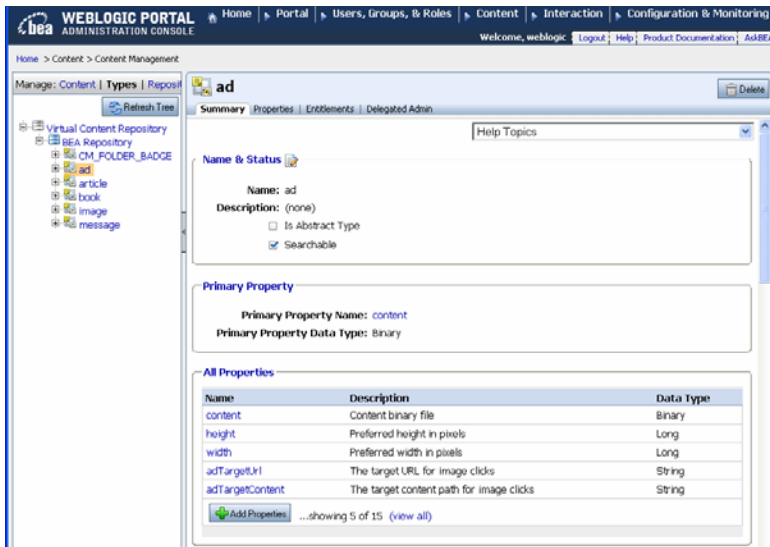
Content Property	Type	Description
adMap	String	Supplies the XHTML definition of an image map. If you specify a value for adMap, you must also specify a value for adMapName.
adWinTarget	String	Displays the target in a new pop-up window, using JavaScript to define the pop-up window. The only value supported for this attribute is <code>newwindow</code> .
adWinClose	String	Specifies the name of a link that closes a pop-up window. The link appears at the end of the window content. For example, if you provide <code>Close this window</code> as the value for this attribute, then “Close this window” appears as a hyperlink in the last line of the pop-up window. If a visitor clicks the link, the window closes.
adAltText	String	Specifies a text string for the <code>alt</code> attribute of the <code></code> tag. If you do not include this attribute, the <code></code> tag does not specify an <code>alt</code> attribute.
adBorder	Integer	Specifies the value for the <code>border</code> attribute of the <code></code> tag. If you do not include this attribute, the <code>border</code> attribute is given a value of zero.

Define the Properties of a Content Type

After creating a content type, you need to define the content properties for that type.

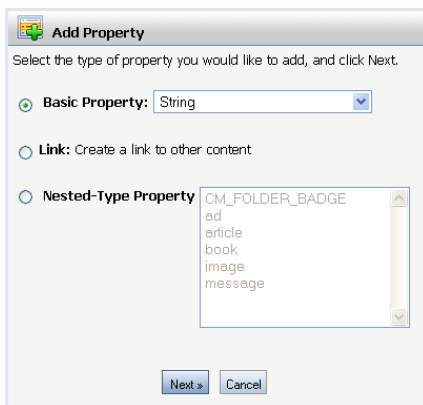
1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Types**.
3. In the resources tree, select the content type to which you want to add a property. See [Figure 6-5](#).

Figure 6-5 The Types Tab within the Manage Types Window



4. Click **Add Property**. See [Figure 6-6](#).

Figure 6-6 Add Property Dialog



5. Select the type of property you want to add: **Basic Property**, **Link** or **Nested-Type Property**.
- To add a **Basic Property**, select a data type from the drop-down list. For more information about data types, see [“Supported Data Types” on page 6-7](#).
 - To add a **Link** property, select **Link**.

- To add a **Nested-Type Property**, select a content type from the list.

Note: Nested type properties allow you to include another content type within the type you are creating. When you do this, all properties from the nested property type are added to the content type you are creating.

6. Click **Next**. The Add Property Dialog appears, as shown in [Figure 6-7](#).

Figure 6-7 Add Property Dialog

7. Enter a name for the property you are creating.
8. Select the property options you want to assign to this property. For more information about these options, see [“Property Options”](#) on page 6-9.
9. If you want this property to hold more than one value, select **Allow Multiple Values**.
10. If you want users to select from a list of values that you provide, select **Restrict Values**. This means they can only enter a value which is included on the list you provide.
11. To add values, click **Add Value Choices**.
12. In the Value Choices dialog, enter the values you need and, if required, select a default value.
13. Click **Save**.

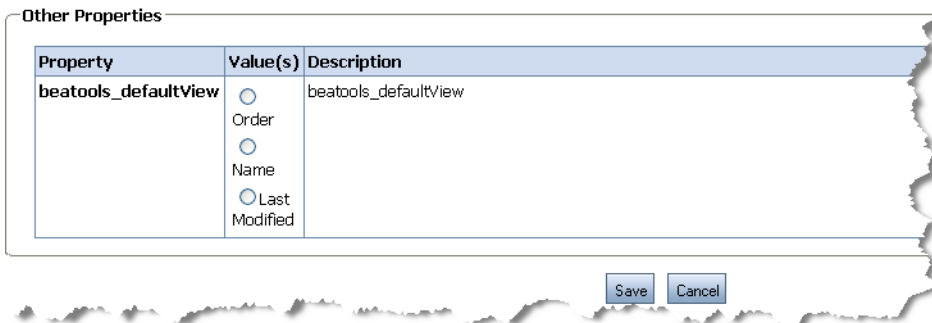
Re-ordering Content Within a Folder Using Properties

You can change the default order used to display content within a folder by modifying the `beatools_defaultview` property of a folder's content type.

To change the default order used to display content within a particular folder:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. In the resources tree, select the folder that you want to modify.
4. Click the **Summary** tab.
5. If using library services, click **Check Out** in the Versioning & Workflow section.
6. Click the **Properties** tab.
7. For the `beatools_defaultview` property, select the **Edit** check box and click **Edit**.
8. In the Other Properties section, select a new value for the `beatools_defaultview` as shown in [Figure 6-8](#).
 - **Order** allows users to use a custom order; see [“Re-Ordering Content”](#) on page 10-11.
 - **Name** sorts the content in alphabetical order by name.
 - **Last Modified** sorts the content by the date it was last modified.

Figure 6-8 Available Property Values for the `beatools_defaultview` Property



The screenshot shows a dialog box titled "Other Properties" with a table containing one row for the property "beatools_defaultview". The table has three columns: "Property", "Value(s)", and "Description". The "Value(s)" column contains three radio button options: "Order", "Name", and "Last Modified". The "Description" column contains the text "beatools_defaultview". Below the table are "Save" and "Cancel" buttons.

Property	Value(s)	Description
beatools_defaultview	<input type="radio"/> Order <input type="radio"/> Name <input type="radio"/> Last Modified	beatools_defaultview

9. When finished making your selection, click **Save**.

10. If using library services, check in your changes.

Out-of-the-Box Content Types

The BEA repository includes five ready-to-use content types. You can use these content types when you add content to your repository or you can create your own.

Although the provided content types may not meet all of your needs, they do provide examples of the properties you can include in your own custom types. In some cases, such as the ad content type, the properties used are required to take full advantage of interaction management features. For more information about interaction management, see the [WebLogic Portal Interaction Management Guide](#).

The following are included content types and the corresponding property definitions. You can re-create these properties in your own types.

- [Ad Content Type](#)
- [Article Content Type](#)
- [Book Content Type](#)
- [Image Content Type](#)
- [Message Content Type](#)
- [CM_FOLDER_BADGE Content Type](#)

Ad Content Type

This content type is used for advertising content. It includes properties that open and close pop-up windows as well as alternative text when the portal user's browser does not support viewing the content. [Table 6-4](#) provides information about the properties included in the ad content type.

Table 6-4 Ad Content Type Properties

Property Name	Data Type	Description
content	Binary	Content binary file.
height	Long	Preferred height, in pixels.
width	Long	Preferred width, in pixels.

Table 6-4 Ad Content Type Properties (Continued)

Property Name	Data Type	Description
adTargetURL	String	The target URL for image clicks.
adWinTarget	String	If set, target render will be in a pop-up window with this name.
adWinClose	String	If set, and adWinTarget set, a close link will be included.
adWinTitle	String	If adWinTarget set, the name of the window.
adClickTarget	String	The target for a click.
adUseXhtml	String	Set to true to produce XHTML, set to false to produce HTML.
adAltText	String	The alternative text for an image.
adMapName	String	The HTML image map name for the image (required if using the adMap property).
adMap	String	The XHTML content for an image.
adBorder	String	The value of the border attribute around the image.
audience	String	Target audience for the content. Default choices are external and internal.

Article Content Type

The article content type is designed to be used for web articles. You can either associate the article file or include the text of the article as a property. Other helpful properties include start and end dates. These dates can be used when configuring campaigns. [Table 6-5](#) details the properties included in the article content type.

Table 6-5 Article Content Type Properties

Property Name	Data Type	Description
contributor	String	Contributor of the article.
startDate	Date	Start date to display the article.

Table 6-5 Article Content Type Properties (Continued)

Property Name	Data Type	Description
description	String	Description of the article.
endDate	Date	Date to stop displaying article.
file	Binary	Binary file of the article.
headline	String	Article headline.
language	String	Language in which the article is written.
articlePriority	String	Article priority.
sizeInBytes	Long	Size of article.
source	String	Source associated with the article.
status	String	Status of the article.
subject	String	Subject of the article.
textContent	String	Text content of the article, if applicable.
url	String	URL associated with the article.
title	String	Article title.

Book Content Type

Use the book content type for books that you may want to suggest to your portal users. The book content type properties allow you to associate sample content from the book as well as author and publication details. [Table 6-6](#) shows the properties included in the book content type.

Table 6-6 Book Content Type Properties

Property Name	Data Type	Description
abstract	Binary	Abstract of the book.
application	String	Application area of the book.
author	String	Author of the book.

Table 6-6 Book Content Type Properties (Continued)

Property Name	Data Type	Description
city	String	City of publication.
country	String	Country of publication.
edition	String	Book edition.
email	String	Publisher e-mail information.
file	Binary	Sample chapter or any file associated with book.
isbn	String	ISBN number of the book.
keywords	String	Keywords associated with book.
link	String	Link associated with book.
note	String	Comments associated with book.
pub_date	Date	Date the book was published.
state	String	State the book was published.

Image Content Type

The image content type can be associated with simple image content. [Table 6-7](#) shows the properties for the image content type.

Table 6-7 Image Content Type Properties

Property Name	Data Type	Description
image_name	String	Name of image.
file	Binary	Image file.
description	String	Description of the image.

Message Content Type

Use the message content type to store message content such as alerts. [Table 6-8](#) provides information about the properties included in the message content type.

Table 6-8 Message Content Type Properties

Property Name	Data Type	Description
subject	String	Subject of message.
date	Date	Date the message was posted.
from	String	Message originator.
organization	String	Organization associated with message.
group	String	Group associated with message.
id	String	Message ID.
body	String	Body text of message.
attachment	String	Attachments to message, usually a binary file.
to	String	Message destination.
message_name	String	Name of message.
link	String	Link to related URL.

CM_FOLDER_BADGE Content Type

The CM_FOLDER_BADGE content type is automatically associated with any folders you create. The properties of this type define the content as a folder instead of a content item. If you want to create content types that are associated with folders, those content types must inherit from the CM_FOLDER_BADGE content type. For more information about type inheritance, see [“Using Content Type Inheritance” on page 6-3](#).

Using Content Types in Your BEA Repository

Using WebDAV with Your BEA Repository

WebDAV is a protocol that allows users to collaboratively edit and manage files on web servers. When you configure WebDAV for your BEA repository, content contributors can save files directly to the BEA repository from Windows Explorer and other Microsoft applications.

Note: For more information about WebDAV, see the [WebDAV web site](#).

This chapter includes the following sections:

- [WebDAV Overview](#)
- [Enabling WebDAV for Repositories](#)
- [Using Content Types with WebDAV](#)
- [Using WebDAV with Your BEA Repository](#)

WebDAV Overview

Adding WebDAV support to your BEA repository effectively adds a new suite of applications to access, create and update content. Content contributors can add files to a BEA repository using Windows Explorer or from Microsoft Office applications.

Before adding WebDAV support to your repository, please review the following:

- [WebDAV Guidelines](#)
- [Supported Versions of Microsoft Office](#)

WebDAV Guidelines

When adding content to your repository using WebDAV, use the following guidelines:

- WebDAV is primarily targeted towards Visitor Entitlement users. Visitor entitlement must be enterprise-application scoped. For more information, see [Using Web-Application or Enterprise-Application Scoped Roles for Entitlements on Portal Resources](#) in the *WebLogic Portal Security Guide*.
- System Administrators can add or modify content.
- Delegated Administration and Portal Administration users cannot add or modify content.
- After adding a file to the repository, you cannot rename it without also renaming the associated file. The name of the content and the associated binary must be the same.
- You can only add content to an existing folder.
- You cannot rename a file and move it at the same time. You must first move the file, then rename.
- As with any content repository, if versioning is not being used, it is possible for content contributors to overwrite work.
- WebDAV utilizes memory when adding files to your repository. For this reason, you should add very large files through the WebLogic Portal Administration Console.

Supported Versions of Microsoft Office

In WebLogic Portal, the implementation of WebDAV requires Microsoft Office 2000, SP3 or greater, with MSDAIIPP.dll version 8.103.3521.0. For additional details about Microsoft's support for WebDAV, see <http://www.greenbytes.de/tech/webdav/webfolder-client-list.html>.

Enabling WebDAV for Repositories

BEA repositories are WebDAV-enabled by default, but can be disabled if you choose. If you want to use WebDAV with a non-BEA repository, you need to enable it.

Note: You must also add a WebDAV content type to your repository, see [“Using Content Types with WebDAV”](#) on page 7-3.

Enabling WebDAV for a non-BEA Repository

BEA repositories are WebDAV-enabled by default. If you want to enable a non-BEA repository to use WebDAV, you need to add a WebDAV property to your repository.

To enable WebDAV for a non-BEA repository within your Virtual Content Repository:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, click the repository for which you want to enable WebDAV.
4. In the Properties section, click **Add Property**.
 - a. In the Name field, enter `WEBDAV_ENABLED`.
 - b. In the Value field, enter `true`.
 - c. Click **Save**.

Disabling WebDAV

If you want to disable WebDAV for a repository, do the following:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repository**.
3. In the resources tree, click the repository for which you want to disable WebDAV.
4. Click **Repository Details**.
5. For the `WEBDAV_ENABLED` property, enter a value of `false`.
6. Click **Save**.

Using Content Types with WebDAV

When users add content to a repository using WebDAV (by using Internet Explorer, Microsoft Word, and so on), content is automatically associated with a content type that defines the metadata required for that content. Optionally, you can define a content type that will automatically pull in document metadata from your Microsoft documents.

How WebDAV Determines Which Content Type to Use

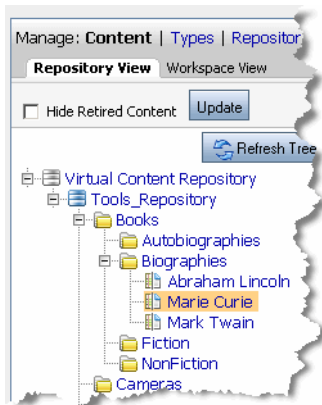
When a user adds a file to your repository using WebDAV, WebDAV associates the content the content type associated with the repository folder where the content is saved.

If no content type is associated with the immediate folder, WebDAV iterates up the resources tree until it finds the first content type associated with a folder. If no content type is associated with any content folder WebDAV finds, it uses the content type associated with the repository. In the example shown in [Figure 7-1](#), if no content type is associated with the Biographies folder, content added to that folder would use the content type associated with the Books folder.

Note: If you are using a file system repository, you cannot associate content types with folders. Therefore, when using a file system repository with WebDAV, all content added to the repository uses the default content type.

If your content type has been defined to be compatible with Microsoft document properties, WebDAV automatically transfers those property values in addition to system properties (such as created date and so on). However users need to log in to the WebLogic Portal Administration Console to check in content or to complete additional administrator-defined properties.

Figure 7-1 Using Content Types with Folders for WebDAV Content



For more information about associating a content type with a folder, see [“Creating a Folder” on page 4-2](#).

Defining a WebDAV Content Type

You should define a default content type to be used by WebDAV when users add content. Although you can assign content types at the folder level that can be used to associate metadata with WebDAV-added content, a repository-wide WebDAV content type assures that content will always be associated with a content type.

To define a default content type for WebDAV content, do the following:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repository**.
3. In the resources tree, click the appropriate repository.
4. In the Properties section, click **Add Property**.
 - a. In the Name field, enter `WEBDAV_TYPE`.
 - b. In the Value field, enter the name of the new content type you want to use. [“Using Content Types with WebDAV” on page 7-3](#)
 - c. Click **Save**.

Giving Users Permission to Create Folders

By default only administrators can create a folder using WebDAV. To allow non-administrators the ability to create folders, you need to add the `CAN_VIEW` and `CAN_INSTANTIATE` capabilities to the `CM_FOLDER_BADGE` type for the given role.

To enable users the ability to create folders using WebDAV:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Types**.
3. In the resources tree, click the `CM_FOLDER_BADGE` type.
4. Click the **Entitlements** tab.
5. If the roles does not exist, add the role, and then go to [step 7](#). For information about adding a role, see [Configuring Visitor Entitlements](#) in the *WebLogic Portal Security Guide*.

6. If the role already exists, click the **Edit Capabilities** check box for the role you want to edit and click **Edit**.
7. In the Entitle Capabilities to Resource dialog, select the **View** and **Instantiate** check boxes.
8. Click **Save**. A summary of the dialog is shown in [Figure 7-2](#).

Figure 7-2 Add View and Instantiate Capabilities

Browse Roles Entitled to this Resource

Showing 1 of 1 ◀ Previous | Next ▶ Items per page 10

Role	Capabilities						Edit Capabilities	Remove Role
	Create	View	Update	Delete	Instantiate	Assign Workflow		
Operator		<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>
Add Role							Edit	Remove

Showing 1 of 1 ◀ Previous | Next ▶ Items per page 10

Creating a Content Type That Maps to Microsoft Document Properties

When adding content directly from Microsoft Office programs, you can configure your repository to automatically fill in metadata using property values from Microsoft document properties. For example, Microsoft Word documents can have properties like Title, Subject, and Keywords. When these properties are present, WebDAV imports them into your repository and associates them with the added content file.

Note: Microsoft document property values override any default values you may have set for that content type.

To use this feature, the WebDAV content type for your repository must include properties that match the Microsoft document properties. You must either add properties to your WebDAV content type or map existing content type property names to the Microsoft property names. For a list of property names, see [Table 7-1](#). For more information about creating a content type, see [“Creating a Content Type” on page 6-4](#).

Limitations

You cannot add multi-valued properties to your Microsoft content using WebDAV, such as lists of keywords. All property values must be defined as single-value.

In addition, you cannot delete property values when you update Microsoft content using WebDAV. For example, if you add a property value to a Microsoft document and then add document to your repository using WebDAV, you must use the WebLogic Portal Administration Console to delete that property value. If you delete the property value in the Document Properties dialog, and then re-add the content to your repository using WebDAV, the property will not be deleted in your repository.

Note: Not all document properties listed in [Table 7-1](#) apply to every kind of Microsoft document. For example, some properties apply to Microsoft Power Point documents but not to Microsoft Word documents.

Table 7-1 Standard Microsoft Document Properties Supported by WebDAV

Name	Description	Data Type
MSO_AUTHOR	Document author.	String
MSO_TITLE	Document title.	String
MSO_CREATE_DATE_TIME	Date and time when document was created.	Calendar
MSO_LAST_MODIFY_AUTHOR	Author who last modified the document.	String
MSO_APPLICATION_NAME	Application which created the document, such as Microsoft Word, Microsoft Excel, and so on.	String
MSO_PAGE_COUNT	Number of pages in the document.	Long
MSO_LAST_MODIFY_DATE	Date when document was last modified.	Calendar
MSO_KEYWORDS	Keywords in the document.	String
MSO_COMMENTS	User comments.	String
MSO_SUBJECT	Subject.	String

Table 7-1 Standard Microsoft Document Properties Supported by WebDAV (Continued)

Name	Description	Data Type
MSO_LAST_PRINT_DATE_TIME	Last time the document was printed.	Calendar
MSO_REVISION_NUMBER	Document revision number.	String
MSO_WORD_COUNT	Document word count.	Long
MSO_TEMPLATE_NAME	Template which created this document. Used by Word documents. Usually normal.dot unless a custom template is used.	String
MSO_MANAGER	Manager field.	String
MSO_LINE_COUNT	Number of lines in the document.	Long
MSO_CATEGORY	Category field.	String
MSO_COMPANY	Company field.	String
MSO_SLIDE_COUNT	Number of slides in the Power Point presentation.	Long
MSO_PARAGRAPH_COUNT	Number of paragraphs in the document.	Long

Mapping Microsoft Properties to Existing Content Type Properties

If you do not want to use Microsoft property names in your content type, you can map a content type property to a Microsoft property by modifying your repository settings.

For example, you may have a content type that includes a property called “Title”. To use this property to collect metadata from WebDAV-added Microsoft documents, you would map the MSO_TITLE property to the “Title” property within your content type.

Note: The data types of the properties must match. For example, if the Microsoft property has a data type of string, the content type property it maps to must also be a string data type.

To map Microsoft document properties to properties within your WebDAV content type:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**
3. In the resources tree, select the appropriate repository.
4. In the Properties section, click **Add Property**.
 - a. In the Name field, enter the name of the Microsoft document property you want to map to a content type property name.
 - b. In the Value field, enter the name of the existing content type property to which you want to map.
 - c. Repeat these steps for each Microsoft property you want to map to an existing content type property.

Using Custom Microsoft Document Properties

You can collect custom metadata for Microsoft documents.

For example, within Microsoft Word, users can add custom properties to documents using the Custom tab in the Properties dialog. If these custom properties are also defined in your WebDAV content type or mapped to an existing content type property, they are automatically associated with the content. If the custom properties are not defined or mapped, do one of the following:

- Add a corresponding property to your WebDAV content type that matches the Microsoft document property name and data type.
- Map the Microsoft property to an existing type, as described in [“Mapping Microsoft Properties to Existing Content Type Properties”](#) on page 7-8.

Using WebDAV with Your BEA Repository

Content contributors must configure their individual environments to take advantage of WebDAV features. Supported applications include Windows Explorer and Microsoft Office. The following procedures enable all WebDAV capabilities that are supported by WebLogic Portal:

- [Enabling WebDAV for an Environment](#)
- [Adding a Microsoft Word Document to a BEA Repository](#)

- [Using Windows Explorer to Add a File to the BEA Repository](#)

Enabling WebDAV for an Environment

To use WebDAV from their local machine, content contributors must enable their environments to recognize the repository as a web location to which to save files.

To enable WebDAV for an individual environment,

1. Open Internet Explorer.
2. Choose **File > Open** to view the Open dialog.
3. In the Open dialog, enter the URL where WebDAV is running. For example, http://<host_machine>:<port>/<portalEARName>Webdav.
4. Select the **Open as Web Folder** check box.
5. Click **OK**.
6. You are asked to login, enter your portal login name and password.

The BEA Repository is listed as a folder within Windows Explorer. You can use Windows Explorer to browse the content in BEA repository just like any other folder.

Adding a Microsoft Word Document to a BEA Repository

If you have configured your local environment to use WebDav, you can save files directly to your repository from Microsoft Office programs. This example explains how to add a Microsoft Word document to a BEA repository.

Note: When adding content to a BEA repository, you need to follow the WebDAV guidelines, see [“WebDAV Guidelines” on page 7-2](#).

1. Within Microsoft Word, choose **File > Save As**.
2. Within the **Save As** dialog, navigate to the location of your BEA repository.
3. Click **Save**.
4. If you have not already logged in, you are prompted to log in. Use the user name and password you use to log into the WebLogic Portal Administration Console.

The file is added to the repository using the default content type with the default values and system properties, such as the version number and date, automatically filled in. For more information, see [“Using WebDAV with Your BEA Repository” on page 7-9](#).

After adding a file to the repository, log in to the WebLogic Portal Administration Console and use the Virtual Content Repository to associate additional content properties for the file you just added. For more information about adding content properties, see [“Adding Content” on page 10-5](#).

Using Windows Explorer to Add a File to the BEA Repository

You can add files to your BEA repository using drag-and-drop in Windows Explorer.

Note: When adding content to a BEA repository, follow the WebDAV guidelines, as described in [“WebDAV Guidelines” on page 7-2](#).

Using WebDAV with Your BEA Repository

Connecting to a Third-Party Repository

The Virtual Content Repository allows you to connect to non-BEA repositories. When third-party repositories are connected to the Virtual Content Repository, their content can be accessed by portal tools such as content placeholders, content selectors, and so on.

Some third-party content management vendors have built integrations (Content Service Provider Implementations or SPIs) that allow you to connect third-party repositories to the Virtual Content Repository. Contact your third-party repository vendor to find out the details about their implementation.

If the third-party repository you are using is JSR 170 compliant, you can connect to it using BEA's JSR 170 Connector> See [“Working with a JSR 170-Compatible Repository”](#) on page 8-6. For more information about JSR170, see the [JSR 170](#) web site.

Note: You cannot use BEA's library services with third-party repositories.

If your third-party repository does not have a written an implementation for the Virtual Content Repository, you can write your own using BEA's Service Provider Interface (SPI). For more information, see the [Content Management SPI Development Guide](#).

This chapter includes the following sections:

- [Working with Third-Party Repositories](#)
- [Working with a JSR 170-Compatible Repository](#)

Working with Third-Party Repositories

Configuring a third-party repository to use with WebLogic Portal involves the following three steps:

1. Create or obtain an SPI implementation for accessing the third-party repository you want to use that integrates the functionality of your third-party repository with WebLogic Portal, as described in the [Content Management SPI Development Guide](#). If you are connecting to a JSR 170-compatible repository, you do not need to write an SPI implementation.
2. Add the repository SPI JARs to your portal classpath to ensure your portal can communicate with the repository. For more information about adding a class to your classpath, see the [WebLogic Server documentation](#).
3. Connect the third-party repository to the Virtual Content Repository using the WebLogic Portal Administration Console. Depending on your implementation, to connect with the Virtual Content Repository, you may need to set various repository properties for your repository. These repository properties are SPI-specific and should be described in the SPI documentation.

Note: You cannot use BEA's library services with third-party repositories.

Connecting to a Third-Party Repository

After creating an SPI implementation, you can connect your third-party repository to the Virtual Content Repository using WebLogic Portal Administration Console.

Tip: If you want to test your repository connection, you can use the `IVirtualRepositoryManager` API.

Once you have connected a third-party repository to the Virtual Content Repository, you can use that repository's content within your portal. If the third-party repository implementation includes write capabilities, you can also use the WebLogic Portal Administration Console to modify content within the repository.

An SPI can be deployed multiple times with different configuration parameters. From the application's perspective, this appears as multiple repositories.

Note: Library services cannot be used in conjunction with SPI implementations.

When you connect to a third-party repository, you may need to configure additional properties that match your third-party repository's configuration. Consult your third-party documentation to verify the properties that you need to configure and the connection class you should use.

To connect a repository to the Virtual Content Repository (using the WebLogic Portal Administration Console):

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, select the **Virtual Content Repository**.
4. On the Browse tab, click **Add Repository Connection**. See [Figure 8-1](#).

Figure 8-1 Browse Tab within the Manage Repositories Window



5. In the Add Repository Connection dialog, provide the following information:

Table 8-1 Repository Connection Information

Field	Description
Name	The name you give your new repository. For example: <code>MyNewRepository</code>
Connection Class	Use the SPI connection class you have created or that has been provided by your third-party vendor. The connection class is the fully qualified name of the class which implements <code>com.bea.content.spi.flexspi.Repository</code> . Be sure you have added this class to your application classpath.
Datasource JNDI Name	Not used for third-party repositories.
Username	If the SPI implementation requires a global username, enter it here. When configuring a BEA repository, you can leave this blank.
Password	If the SPI implementation requires a global password, enter it here. When configuring a BEA repository, you can leave this blank. Note: For information about configuring a cleartext password for a third-party repository, see “Setting User Credentials for an External Repository in content-config.xml” on page 14-2 .
Retype Password	If you entered a global password, re-enter it here.
Enable Library Services	Not used for third-party repositories.

- Click **Save**.
- Within the Repositories section, click the repository you just created to verify that it has been created and view its Repository Summary.

Logging Into a Third-Party Repository

If the third-party repository connected to BEA’s Virtual Content Repository supports multiple user authentication, you can restrict users to specific content. Multiple user authentication lets users log in with global credentials or with unique user credentials. User credentials can control the content individual WebLogic Portal users can view. For example, content contributors could view only the content that they add or edit.

Repositories, such as SharePoint, that support the `RepositoryMultipleUsers` capability type can display the Login Options button. Multiple user authentication is available for content repositories that support the `RepositoryMultipleUsers` capability type, defined by the `RepositoryFeatureCapability.RepositoryMultipleUsers` (which is determined by the `ICapabilityManager.checkRepositoryCapability()` method).

The Login Options button appears automatically in the WebLogic Portal Administration Console if you connected the third-party repository to BEA's Virtual Content Repository.

Caution: Currently there are no third-party repositories that support multiple user authentication.

Perform the following steps to set up global credentials and log into the repository:

1. After you connect to a third-party repository that supports multiple user authentication, start the WebLogic Portal Administration Console.
2. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
3. To set up global credentials, perform the following steps:
 - a. Click **Manage | Repositories**.
 - b. In the resources tree, select the repository.
 - c. In the Summary tab, click **Change Password**.
 - d. Enter a global user name, password, password confirmation, and then click **Save**.
4. To log into the third-party repository, click the **Manage | Content** tab and select the third-party repository. After you select the repository, the Login Options button appears, as shown in [Figure 8-2](#).

Figure 8-2 The Login Options Button



5. Click **Login Options**, select one of the following:
 - Click **Login with Global Credentials** to log in with the global credentials that you set up earlier. All content in the third-party repository is visible.
 - Click **Login with my saved credentials for user xyz** to log in to see the content this user is authorized to view.

- Click **Update my credentials and login** to enter a new user name and password. You must have logged in at least once to select this option.

6. Click Login. See [Figure 8-3](#).

Figure 8-3 Select Global or User Credentials to Log into the Third-Party Repository

Login for Repository: SharePointRepository

Login with global credentials

Login with my saved credentials for user cmsca

Update my credentials and login

Username:

New Password:

Retype New Password:

Note: this login setting will be saved until you update it again.

Currently logged in as user: cmsca

User authentication persists across multiple portal sessions until you change it.

Working with a JSR 170-Compatible Repository

WebLogic Portal provides a connector to repositories that implement the JSR 170 specification such as Day Software’s CRX. If you want to access a repository using its JSR 170 interface, you do not need to write a custom SPI implementation.

This section assumes you have already installed and configured a JSR 170 repository. For additional documentation about installing and using Day Software’s CRX JSR 170 repository, see the [WebLogic JSR 170 Adaptor Developer’s Guide](#) and the [WebLogic JSR 170 Supported Configurations Guide](#).

Searching within a JSR 170 Repository

When a repository is connected to the Virtual Content Repository, both content contributors and developers can search for content within the repository. However not all metadata provided by the Virtual Content Repository, such as system properties and MIME types, are supported by the JSR 170 specification; subsequently some searches may be invalid.

The following system properties cannot be used when searching JSR 170 repositories:

- `cm_binaryName`

- cm_nodeName
- cm_createdBy
- cm_modifiedBy
- cm_createdDate
- cm_modifiedDate
- cm_contentType
- cm_binarySize

The following search operators cannot be used:

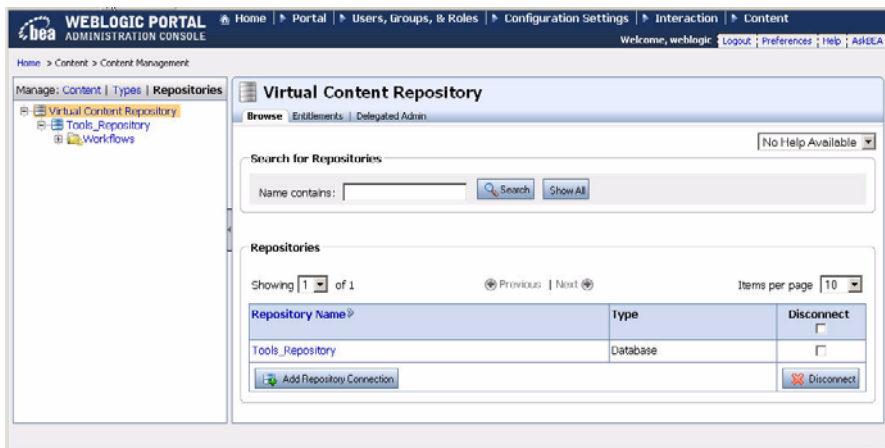
- containsall
- likeignorecase

Connecting to a JSR 170 Repository

To connect to a JSR 170 repository, do the following:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the Repositories tree, click the **Virtual Content Repository**.
4. In the Browse tab, click **Add Repository Connection**. See [Figure 8-4](#).

Figure 8-4 Browse Tab within the Manage Repositories Window



5. In the Add Repository Connection dialog, provide the information shown in [Table 8-2](#):

Table 8-2 Add Repository Connection

Field	Description
Name	The name you give your new repository.
Connection Class	The name of the connector class which connects the repository to WebLogic Portal. If connecting to the Day Software CRX repository, use the following connection class: <code>com.day.content.spi.jsr170.JNDIRepository</code>

Table 8-2 Add Repository Connection (Continued)

Field	Description
Datasource JNDI Name	Used only for BEA repositories.
Username	Enter the user name required to connect to your third-party repository.
Password	Enter the password required to connect to your third-party repository. Note: For information about configuring a cleartext password for a third-party repository, see “Setting User Credentials for an External Repository in content-config.xml” on page 14-2.
Retype Password	Re-enter the password required to connect to your third-party repository.
Enable Library Services	Used only for BEA repositories. Deselect this check box to ensure library services are disabled. Note: Library services are not supported for third-party repositories.

6. Click **Save**.
7. Click the repository you just created to verify that it has been created and view its Repository Summary.
8. In the Properties section, click **Add Property** and add the properties in [Table 8-3](#) to your repository.

Table 8-3 JSR170 Connector Properties

Repository Property	Required Value	What it does
<code>jsr170.hide.builtin</code>	true	Prevents repository-specific types and properties from interfering with the Virtual Content Repository.
<code>jsr170.uid.mapping</code>	uuid	Maps the IDs generated by your repository to the internal IDs needed by the Virtual Content Repository.

9. In the Properties section, verify that the property has been created.

Connecting to a Third-Party Repository

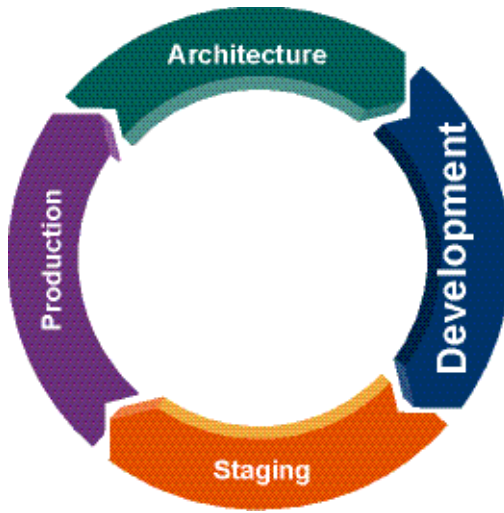
Part II Development

Within content management, the development phase covers both creation of content as well as delivering that content to your portal users. Using the WebLogic Portal Administration Console, content contributors add content to the content repository using workflows and versioning. Using WorkSpace Studio, developers use to incorporate content within the portal by using personalization tools such as content selectors and placeholders. Developers can also display content with JSP tags, controls and the content API.

Part II includes the following chapters:

- [Chapter 9, “Delivering Content Within Your Portal”](#)
- [Chapter 10, “Adding Content to a BEA Repository”](#)
- [Chapter 11, “Using Display Templates”](#)
- [Chapter 12, “Using Syndicated Feeds”](#)

For a detailed description of the development phase of the portal life cycle, see the [BEA WebLogic Portal Overview](#). The portal life cycle is shown in the following figure:



Delivering Content Within Your Portal

When developing your portal, you use JSP tags, the content API, and personalization tools to retrieve and display content to your portal users.

Retrieving and displaying content is typically done within a JSP page. Within a JSP page, you can use the API, JSP tags, content selectors, placeholders, and campaigns to retrieve content based on queries and personalization rules.

For more information about delivering personalized content, see the following sections in the *WebLogic Portal Interaction Management Guide*:

- [Creating a Content Selector](#)
- [Creating a Placeholder](#)
- [Building a Campaign](#)

This chapter includes the following sections:

- [Working with JSP Tags](#)
- [Displaying Content with JSP Tags](#)

Working with JSP Tags

JSP tags both retrieve and display content. Retrieving content or searching for the content is typically done within the context of a JSP page. You can also retrieve content using the content API.

Retrieving Content with JSP Tags

The four content-specific JSP tags that retrieve content are listed in [Table 9-1](#). For more information about JSP tags, see the [JSP Javadoc for Content Management](#).

Table 9-1 Content JSP Tags

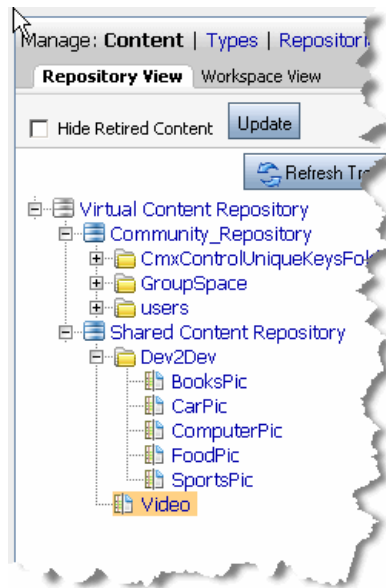
JSP Tag	Usage
<code><cm:getNode></code>	Retrieves a content node and stores the node in a variable.
<code><cm:search></code>	Searches for and retrieves content nodes based on a supplied query and stores the results in a variable.
<code><cm:getProperty></code>	Retrieves a property value from a content node and stores it as a variable or prints it in the JSP.
<code><cm:getBinaryProperty></code>	Retrieves a binary property associated with a content node.

Regardless of which JSP tag you use, you retrieve content within a tag by using a repository path or by using queries.

Retrieving Content Using a Repository Path

All content is addressable by a unique path. This path is visible in the WebLogic Portal Administration Console as you create folders and content. Within the `<cm:getNode>` JSP tag, you can specify the repository path to retrieve content. For example, if your content hierarchy appears in the WebLogic Portal Administration Console as shown in [Figure 9-1](#), you could use the code in [Listing 9-1](#) to retrieve `CarPic`.

Figure 9-1 Content Hierarchy



Listing 9-1 Sample Code for Retrieving CarPic

```
<%@ taglib uri="content.tld" prefix="cm"%>
<cm:getNode path="/BEA Repository/Dev2Dev/CarPic" id="carpic" />
```

Using Queries to Retrieve Content

You can also use queries to retrieve content. For more information about creating queries, see [Building a Content Query with Expressions](#) in the *Interaction Management Guide*.

Displaying Content with JSP Tags

After retrieving content, you need to decide how to display it within the portal. This section discusses using JSP tags and provides some examples. Displaying content can also be done using personalization tools. For more information about personalizing content, see the *Interaction Management Guide*.

[Table 9-2](#) list the JSP tags that can display content. For a complete reference for the tags listed, see the [JSP Javadoc for Content Management](#).

Table 9-2 JSP Tags Used to Display Content

JSP Tag	Description
<ad:adTarget>	Uses the Ad service to send an ad query to the content management system.
<ad:render>	Renders a content node from the Virtual Content Repository in the current JSP.
<ph:placeholder>	Displays personalized web content in a JSP based on placeholder and campaign rules and queries that have been defined.
<pz:contentQuery>	Performs a content attribute search in a content management system and returns an array of content objects.
<pz:contentSelector>	Implements a Content Selector that has been defined using Workspace Studio. Displays personalized Web content in a JSP based on Content Selector rules and queries defined.
<pz:div>	Allows a piece of in-line content to be shown if a user belongs to the specific User Segment defined with Workspace Studio.

Table 9-2 JSP Tags Used to Display Content

JSP Tag	Description
<code><dt:displaytemplate></code>	<p>Requests the registered view for a resource, which in turn renders it in the JSP.</p> <p>To use this tag, you must create display templates (views) and a mapping file (XML) to map to the templates. For more information about setting up templates, see Chapter 11, “Using Display Templates.”</p>
<code><dt:displaycmtemplate></code>	<p>Requests the registered view for a content management resource, which in turn renders it in the JSP.</p> <p>To use this tag, you must create display templates (views) and a mapping file (XML) to map to the templates with the respective content types. For more information about setting up templates, see Chapter 11, “Using Display Templates.”</p>

Delivering Content Within Your Portal

Adding Content to a BEA Repository

Content contributors create content in the BEA repository through the Virtual Content Repository. The Virtual Content Repository is accessible through the WebLogic Portal Administration Console and provides access to all configured BEA repositories.

Creating content involves associating a content file with a content type and uploading the file to the repository. If your BEA repository takes advantage of library services, it includes the ability to track content versions and use content workflows that enforce a publication process.

This chapter includes the following sections:

- [Viewing the Virtual Content Repository](#)
- [Working with BEA Repository Content When Using Library Services](#)
- [Searching for Content within Your Repository](#)
- [Using Versioning](#)
- [Changing the Workflow of Content](#)

Note: Before content can be created, you must first create content types, design your repository hierarchy, and (if using library services) add content workflows. For more information about these subjects, see:

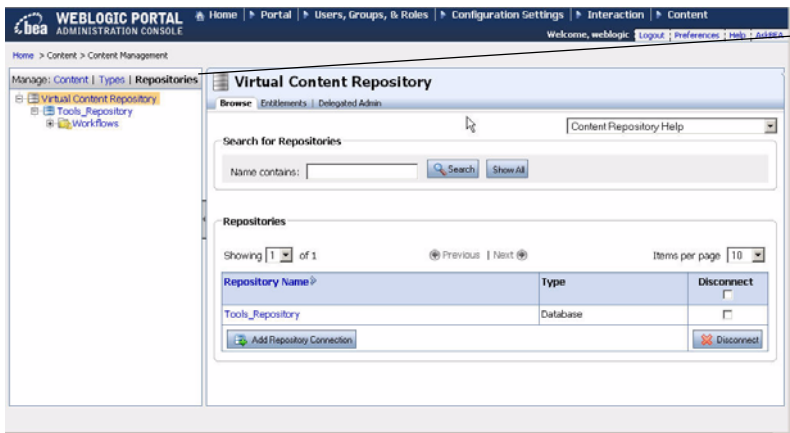
- [Chapter 4, “Using Content Folders in Your BEA Repository”](#)
- [Chapter 5, “Using Content Workflows in Your BEA Repository”](#)
- [Chapter 6, “Using Content Types in Your BEA Repository”](#)

Viewing the Virtual Content Repository

The Virtual Content Repository allows you to view your content repositories in three ways: Content, Types, and Repositories.

The Virtual Content Repository is typically accessed through the WebLogic Portal Administration Console, see [Figure 10-1](#).

Figure 10-1 The Virtual Content Repository within the WebLogic Portal Administration Console



The Virtual Content Repository provides three views: Content, Types, and Repositories.

[Table 10-1](#) lists each view and the tasks you accomplish in each view.

Table 10-1 Summary of the Views of the Virtual Content Repository

Virtual Content Repository View	Associated Tasks
Content view (Click Manage Content)	<ul style="list-style-type: none"> • Add content to the repository. • Search for content. • Delete content. • Modify content. <p>If using BEA's library services, you can also:</p> <ul style="list-style-type: none"> • View version history. • Check out content. • Check in content.
Types view (Click Manage Types)	<ul style="list-style-type: none"> • Add content types. • Modify content types (adding or removing property definitions). • Delete content types.
Repository view (Click Manage Repositories)	<ul style="list-style-type: none"> • Connect repositories to the Virtual Content Repository. • Edit repository properties such as search settings and caches. <p>If using BEA's library services, you can also:</p> <ul style="list-style-type: none"> • Add content workflows. • Modify content workflows. • Delete content workflows.

Working with BEA Repository Content When Using Library Services

Typically, when you use a BEA repository, it is enabled to use library services. Library services provide versioning and content workflows. This chapter assumes your BEA repository uses library services, which means that you must check in content when you create it and check content out before modifying it. Additionally, library services includes content workflows, which means that content must always be assigned a workflow status, such as Draft or Published. For

more information about using content workflows, see [Chapter 5, “Using Content Workflows in Your BEA Repository.”](#)

Note: You can allow or prevent users from being able to create or modify content by using Delegated Administration. For detailed information about setting up Delegated Administration on content resources, see the *WebLogic Portal Security Guide*.

To add content to a repository you can use content management tools in the WebLogic Portal Administration Console or, if WebDAV is enabled, Microsoft Explorer or other Microsoft applications. For information about adding content from these applications, see [“Using WebDAV with Your BEA Repository” on page 7-9](#).

Overview of Library Services

Library services provide the following benefits:

- Workflow management for customizing content development workflows.
- Content workspaces for simplified user-based views of in-progress content.
- Version control for content items and content types.

For instructions on how to enable a BEA repository to use library services, see [“Enabling Library Services for a BEA Repository” on page 3-2](#).

[Table 10-2](#) provides an overview of library services.

Table 10-2 Overview of BEA Repository Library Services

Library Service	What it provides:
Content Workflow	Allows you to move content through a set of status states. The default content workflow includes Draft, Ready for Review, Rejected, Published and Retired. You determine who can change status Delegated Administration.
Content Workspace	<p>Provides a simplified user-based view of content that is currently in progress. It contains two folders: Checked-Out Items and Assigned Items. Checked-Out Items contains all content items that the current user has checked out for edit. Assigned Items contains all items that are assigned to that user and require further action, such as approval.</p> <p>Use the Content Workspace to:</p> <ul style="list-style-type: none"> • View items that are currently assigned to your role. All users within a particular role view and modify items assigned to that role. • View items that you currently have checked out. This folder displays only those items checked out by the current user.
Content Versioning	Allows you to keep track of multiple versions of a content item. Versioning provides a more powerful alternative than backing up files. With content versioning, you can easily view previous versions of a content item if needed, and keep a detailed record of when and why changes were made.

Adding Content

When you add content to your repository, you either upload an existing file to your repository and associate that file with a content type or create HTML content directly in the repository. To create HTML content, see [“Creating HTML Content” on page 10-7](#).

In this example, you create content using the “book” content type, which is one of the default content types included with your BEA repository. Your repository may have other customized content types that differ from this example.

Note: This example assumes you are using a library services-enabled repository. If you are not using library services, you are not prompted to check in your content or select a workflow status.

To add a content file to your repository:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the resources tree, select the repository to which you want to add content.
5. In the Browse Contents, click **Add Content**.
6. In the Add Content dialog:
 - a. Enter a name for the content.
 - b. Select the **book** content type from the Type list.
 - c. Click **Add**.
7. In the Add Properties page:
 - a. In the Primary Properties section, associate content with a file by clicking **Upload File**.
 - b. Click **Browse** to select a content file from your file system.
 - c. If you know the type of encoding you wish to use, select it from the drop-down list. If not, use **Detect from Browser Request**.
 - d. Optionally, select the MIME type that matches the content you selected.
 - e. In the Other Properties section, provide values for the properties listed. These property values are used by portal developers to retrieve your content.
8. To finish do one of the following:
 - Click **Save** to save your content item and not check it into the repository. While content is checked out of the repository, it is displayed in the Workspace View under the Checked-Out Items folder.
 - Click **Save & Check In** to save your content item and check it into the repository.

Creating HTML Content

When adding content that includes a property that allows you to include a binary file, you can use the content editor to create an HTML file. The content editor allows you to create both HTML forms and documents.

To create HTML content:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the resources tree, select the repository to which you want to create content.
5. In the Browse tab, click **Add Content**.
6. In the Add Content dialog:
 - a. Enter a name for the content.
 - b. In the Type list, select the content type you want to associate with the content item.
 - c. Click **Add**.
7. In the Add Properties page under Other Properties, enter the required property values.
8. To add an HTML file using the content editor:
 - a. In the Primary Property section, click **Create Document**.
 - b. In the content editor, click HTML.
 - c. Use the content editor to create an HTML file.
 - d. Click **Save**.
9. In the Add Properties page, click **Save**.

Tip: You can also cut and paste HTML from existing documents (such as Microsoft Word or other applications that save as HTML) into the content editor.

Modifying Content

You can modify repository content by updating the property values associated with the content, updating the associated file, or by changing the workflow associated with the content. When using a library services-enabled repository, you must check out content before you can modify it.

To edit content:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. If you are not using library services, select the content you want to edit in the resources tree and go to [step 7](#).
4. If you are using library services, click the **Workspace View > Assigned Items** folder and select the content you want to edit.
5. In the Versioning & Workflow section of the Summary page, click **Check Out**.
6. In the Checked-Out Items folder, select the content you want to edit.
7. You can edit the content name, its properties, change the binary file associated with the content, or change the workflow associated with the content.
8. To edit the content name:
 - a. Within the Summary page, click **Name & Type**.
 - b. In the Edit Name dialog, enter the new name and click **Update**.
9. To edit a content property:
 - a. In the Properties section, click **Properties**.
 - b. In the Other Properties section, select the property or properties you want to edit by selecting the associated check box in the Edit column.
 - c. Click **Edit** and update the properties as needed.
 - d. To save your content item and not check it into the repository, click **Save**.
 - e. To save your content item and check it into the repository, click **Save & Check In**.
10. Optionally, in the Check In dialog, choose a new workflow status for the content.

Changing the Status of a Single Content Item

You can change the workflow status of a single content item by checking it out and then checking it back in again. For more information about workflows, see [“Creating Content Workflows” on page 5-4](#).

To change the workflow status for content:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Click the **Workspace View > Assigned Items** folder and select the content you want to edit.
4. In the Versioning & Workflow section of the Summary page, click **Check Out**.
5. In the Checked-Out Items folder, select the content for which you want to change status.
6. In the Versioning & Workflow section of the Summary tab, click **Check In**.
7. In the Check In Content dialog, select a new status from the drop-down list.
8. Click **Check In**.

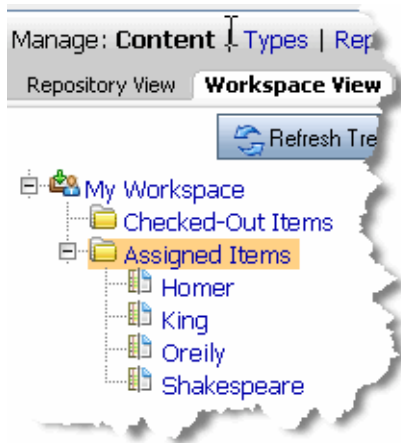
Changing the Workflow Status of Multiple Content Items

You can change the workflow status of multiple content items by performing a bulk update. To update multiple content items at once, each item you want to update must be either checked out or currently assigned to you. When you do a bulk update, each content item you changed is also checked in to the repository during the update.

To change the status of multiple content items at once:

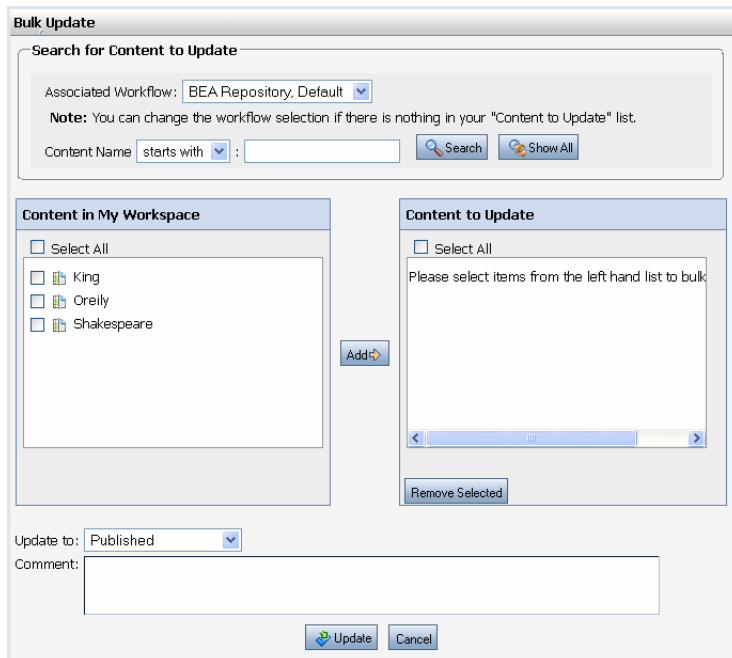
1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Click the **Workspace View > Assigned Items** folder. See [Figure 10-2](#).

Figure 10-2 Assigned Items Folder



4. In the Browse Tab, click **Bulk Update**. See [Figure 10-3](#).

Figure 10-3 Bulk Update



5. In the Search for Content section, select a workflow from the Associated Workflow drop-down list.

Note: The Associated Workflow list only appears if your assigned items use more than one workflow.

6. In the Content in My Workspace section, select the content you wish to update.
7. Click **Add** to move your selections to the Content Update section.
8. Select a new workflow status from the Update to drop-down list.
9. Optionally, add any comments.
10. Click **Update**.

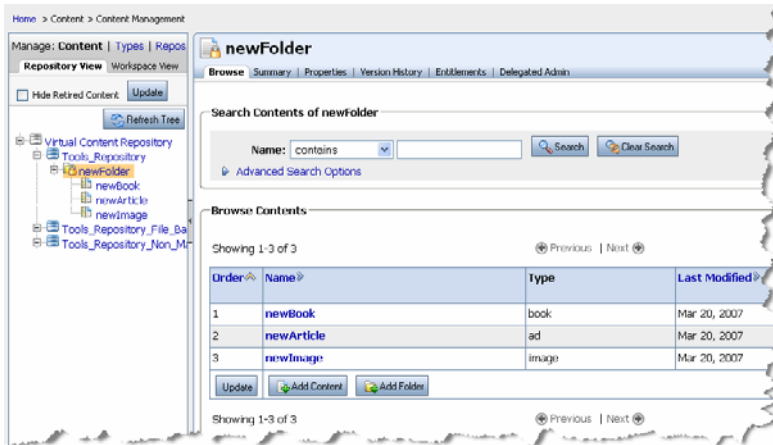
Re-Ordering Content

You can change the order in which content displays in the WebLogic Portal Administration Console.

To re-order content within a folder:

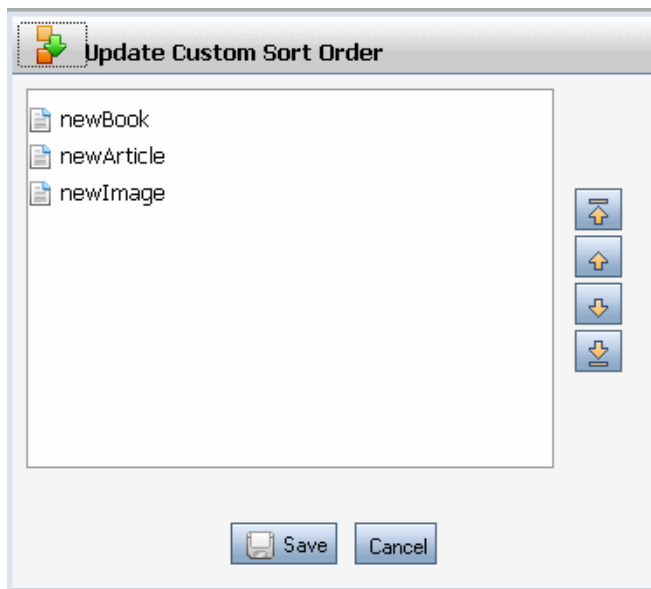
1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Click **Repository View**.
4. Select the folder containing the items you want to re-order.
5. In the Browse Contents section of the Browse tab, click **Order** as shown in [Figure 10-4](#).

Figure 10-4 Re-ordering Content in the Browse Contents Tab



6. Click **Update**. See Figure 10-5.

Figure 10-5 Update Custom Order Dialog



7. Use the navigational arrows to re-order your content and then click **Save**.

Tip: You can also change the default sort order by modifying the content type properties of the folder. For more information, see [“Re-ordering Content Within a Folder Using Properties”](#) on page 6-16.

Updating Binary Content

Binary properties are usually the primary property, although you can have more than one binary property within your content.

To update a binary file:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Workspace View**.
4. In the Assigned Items folder, select the content that you want to update.
5. In the Versioning & Workflow section of the Summary tab, click **Check Out**. The content is moved to the Checked-Out Items folder.
6. In the Checked-Out Items folder, select the content.
7. Select the **Properties** tab.
8. If editing a primary property:
 - a. In the Primary Property section, click **Edit** in the Edit column.
 - b. In the Open dialog, choose to save to disk.
 - c. After copying the file to your local drive, make and save your changes.
 - d. When ready to upload your changes, click **Upload File**.
 - e. Click **Save** or click **Save and Check In**.
 - If you click Save, your content item is saved and remains checked out of the repository.
 - If you click Save and Check In, you can add comments and choose a new workflow status for the content.

9. If editing a property other than the primary property:
 - a. In the Other Properties section, select the **Edit** check boxes for the properties you want to edit, and then click **Edit**.
 - b. Edit the fields for the properties you selected.
 - c. Click **Save** or click **Save and Check In**.
 - If you click Save, your content item is saved and remains checked out of the repository.
 - If you click Save and Check In, you can add comments and choose a new workflow status for the content.

Deleting Content

You can delete content or content folders from your repository at any time. When you delete content, you delete any children content items of the deleted item. If you do not want to delete the children content items of the item you are deleting, use the Move command to move them to another location within the repository before deleting the parent content item. see [“Moving Content” on page 10-15](#).

If your repository is library services-enabled, when you delete content, you delete all versions of that content. If you do not want to delete all versions, you can choose to retire a content version instead. For more information, see [“Retiring Content” on page 10-24](#).

To delete content:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the resources tree, right-click the content item you want to delete and select **Delete** from the menu.
5. In the Delete dialog, click **Delete (<content name>)**.

Moving Content

You can move content or a content folder to another location within the repository. When you move content, you also move any children items.

To move content or a folder:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the resources tree, right-click the content item you want to delete and select **Move** from the menu.
5. Click **OK** in the Move dialog.
6. Right-click on the repository location (folder or content) where you want to copy the content item and choose **Paste (<content name>)**.
7. Click **OK** in the Paste dialog to confirm.

Linking Content

Some content types allow you to link to other content within the Virtual Content Repository. Content links allow you to associated content so it can be easily maintained and updated.

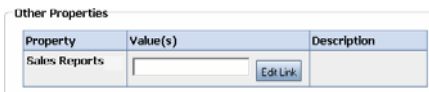
To use a content link, the content type associated with the content must include a link property. For more information about setting up a link property in a content type, see [“Using Link Properties” on page 6-11](#).

Note: When using versioning, you cannot link to content that has not been checked in to the repository at least once. In order to link to content, that content must have a version number of at least 1.

To link content:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Workspace View**.
4. In the Assigned Items folder, select the content that you want to update.
5. In the Versioning & Workflow section of the Summary tab, click **Check Out**. The content is moved to the Checked-Out Items folder.
6. In the Checked-Out Items folder, select the content.
7. Click the **Properties** tab.
8. In the Other Properties section, select the check box for the link property you want to edit, and then click **Edit**.
9. Click **Edit Link**. See [Figure 10-6](#).

Figure 10-6 Edit Link Button



10. Use the controls in the Update Repository dialog to select the linked content.
11. Click **Update Link**.
12. Click **Save** or **Save & Check In**.
 - If you click Save, your content item is saved and remains checked out of the repository.
 - If you click Save and Check In, you can add comments and choose a new workflow status for the content.

When finished, you can see the path to the linked content in the Current Value(s) properties column.

Renaming Content

You can rename content or a content folder in the BEA repository.

Note: For library services-enabled repositories, when you rename content, you do not create a new version of content.

To rename content or a content folder:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the resources tree, right-click the content and select **Rename**.
5. In the **Rename** dialog, enter in the new name for the content item and click **Rename**.

The resources tree now shows the new name for the content.

Copying Content

You can copy content or a content folder to another location in the repository.

Note: For library services-enabled repositories, when you use the **Copy** command, the content item's version history is deleted. If you want to retain version history information, use the Move command, see [“Moving Content” on page 10-15](#).

To copy content or a content folder:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the resources tree, right-click the content and choose **Copy**.
5. Click **OK** in the Copy dialog.
6. Right-click the repository location (folder or content) where you want to place the content item and select **Paste (<content>)**.
7. Click **OK** in the Paste dialog to confirm. The content displays in its new location.

Previewing Content

You can preview binary files associated with content. Previewing binary files allows you to verify that they were uploaded properly. When you preview files, you can choose the application on your hard drive with which to preview the file.

To preview content:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Using either the **Repository View** or the **Workspace View > Assigned Items** folder, click the content you want to edit in the resources tree.
4. If using the Repository View, click to the content that you want to preview in the resources tree.
5. If using the Workspace View, select the content you want to preview in the Checked-Out Items folder.
6. In the **Summary** tab, under Name & Type, click the **Preview** icon.
7. If necessary, select an application with which to preview the content.

Searching for Content within Your Repository

As your repository grows, you can save time by using the Virtual Content Repository's search features to find the content you want to work with. You can search the version history of a particular content item (if using library services), as well as conduct a search of the entire repository. You can also do a full-text search of repository content to search the binary files for keywords or subjects.

Note: You can reorder content in the Browse tab by modifying the Order column providing the folder allows re-ordering of content. For more information, see [“Re-Ordering Content” on page 10-11](#).

Multi-language Searching and Indexing

WebLogic Portal utilizes Autonomy® search for its search functionality. To search in multiple languages or to set a server or repository for a specific language, see [Multi-language Searching and Indexing](#) in *Integrating Search*.

Searching for Content By Name

To search for content by name:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the resources tree, click the repository you want to search.
5. In the right pane, select the **Search Repository** tab.
6. In the Name field, select an operator from the drop-down list and type the name that you wish to use.
7. Click **Name Search**.

Searching for Content By Property Value

You can use the property search when you know the properties used for the content you want to find. For example, if you want to find all content that was published on a certain day, you would search the published date property for a value of the date you want to find.

When you search for content according to property values, you can search published content, unpublished content or all content.

To search your repository for content using property values:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the right pane, select the **Search Repository** tab.
5. Specific the properties for which you want to search. [Table 10-3](#) for specific instructions on each criteria. You can use any or all of the criteria. If a criteria is left blank, it is not used in the search.

Table 10-3 Specific Properties Search Criteria

Search Criteria	Usage
Publish Status	Choose whether to search Published and/or New (unpublished) content.
Search Full Text	You cannot search for properties with this selection.
Or Search Specific Properties	Select to search for properties.
Matches any condition Matches all conditions	Choose whether you want your search to include any of the conditions you specify or all of them by marking the appropriate option.
Type	Selecting a type determines which content type you want to find. Choose the content type(s) you want to search from the list.
Also search types that inherit from this type	Select to search for types that inherit from the type(s) you choose in the Type list.
Type Properties	Define the properties and values for which you want to search. Select a property from the drop-down list, choose an operator, and enter the value you are searching for.
Dates	Select Created or Modified from the drop-down list, select an operator, and use the calendar tool to choose a date.
Creator or Editor	Select Created By or Modified By from the drop-down list and enter a user name in the provided field.

Table 10-3 Specific Properties Search Criteria (Continued)

Search Criteria	Usage
Binary Content	Select the property of the binary file, an operator, and enter a value you want to search.
Path to Content	Path to Content refers to the folder or sub folder that the search content resides in within the repository. Enter the folder name(s) you want to search using the following format: <i>/foldername/subfoldername/contentname</i>

- When finished entering criteria, click **Advanced Search**. Your results display in the Search Results section.

Searching the Full Text of Content and Property Values

You can search the full text of a content item including its property values and associated file. You can also use full-text search to find content of different content types that might include the same information. Full-text search can only find repository content that has been published for delivery to your portal. For information about using full-text with different languages, see [Multi-language Searching and Indexing](#).

Tip: You can use expressions to search for content within your BEA repository. For more information, see “Building a Content Query with Expressions” in [Creating a Content Selector](#) in the *WebLogic Portal Interaction Guide*.

To use a full-text search to find content:

- From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
- Select **Manage | Content**.
- Select **Repository View**.
- In the right pane, select the **Search Repository** tab.
- Click **Advanced Search Options** in the **Search** section.
- If necessary, select the **Published** check box. Full-text search does not work on unpublished content.

7. Mark **Search Full Text**.
8. In the text box provided, enter the keywords for your search. The text you provide will be used to search both the property values for content and the associated binary files.
9. Click **Advanced**
10. **Search**.

Using Versioning

If your BEA repository is library services-enabled, your content is automatically versioned, which means a new copy of content is saved whenever you check in content to the repository. Content versions are numbered and can be searched and managed as other content.

Checked out content is displayed in the Workspace View.

Checking Out Content

If you are using a library services-enabled repository, versioning is enabled. Before you can modify content within the repository, you must check it out.

You can check out content from two locations:

- Use the Repository View if the item is not currently assigned to you.
- Use the Workspace View if the item is currently assigned to you. Items assigned to you appear in the Workspace View under the Assigned Items folder.

To check out content from the Repository View:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the resources tree, click the content you want to check out.
5. In the Versioning & Workflow section of the Summary tab, click **Check Out**.

To check out content from the Workspace View:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.

2. Select **Manage | Content**.
3. Select **Workspace View**.
4. In the Assign Items folder, select the content you want to check out.
5. In the Versioning & Workflow section of the Summary tab, click **Check Out**.

Your content item is now displayed in the **Checked-Out Items** folder. Use the steps listed in [“Modifying Content” on page 10-8](#) to edit your content.

Checking In Content

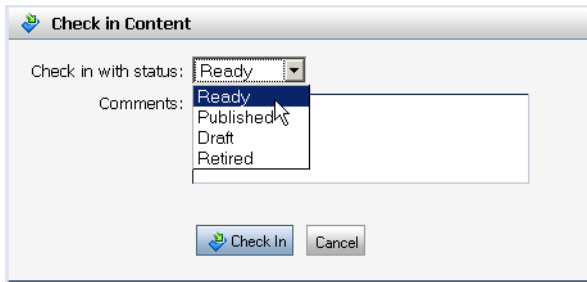
When working with content within a BEA repository that has library services-enabled, you must check in content before it becomes available to other users and portal search queries.

You use the Workspace view to check in content.

To check in content from the Workspace View:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Manage | Content** to view the content resources tree.
4. Select **Workspace View**.
5. In the Checked-Out Items folder, select the content you want to check in.
6. In the Versioning & Workflow section of the Summary tab, click **Check In**.
7. Optionally, in the Check In Content dialog, change the status of the content item and enter a comment. See [Figure 10-7](#).

Figure 10-7 Changing Workflow Status when Checking In Content in the WebLogic Portal Administration Console



Retiring Content

Retiring content means that it can no longer be used in search queries or displayed in your portal. Retiring content offers an alternative to deleting content from your portal. If a repository is library services-enabled, if you delete content, all versions of that content are deleted, see [“Deleting Content” on page 10-14](#).

When you retire content, you change the workflow status of content item to Retired status. This topic assumes you are using the default content workflow. If you are using a custom workflow, the name of the Retired status may be different. Please see your content management administrator for more information.

Note: Content workflows are only available if your repository is library services-enabled.

To retire a content version:

1. Check out the content you want to retire as described in [“Checking Out Content” on page 10-22](#).
2. Check in the content you want to retire as described in [step 1 through step 6 in “Checking In Content” on page 10-23](#).
3. In Check In Content dialog, change the status of the content item to **Retired** and click **Check In**.

Viewing and Searching Version History

You can view the version history for any content item in your repository. Version history information includes the date content was modified, the name of the user who modified the content, and the status of the version. You can search the version history of a content item either by what is contained in the version comments or by using advanced options such as when the content was modified and by whom.

To view the version history of a content item:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Using either the Repository View or the **Workspace View > Assigned Items** folder, click the content whose version history you want to search.

Note: You can also use the Search tab. See [“Searching for Content By Name” on page 10-19](#).

4. Click the **Version History** tab.
5. To search within the version history of a content item:
 - a. To search the version comments for specific text, enter the text you want to find and click **Search**.
 - b. To search the version history by workflow status, date modified, or by the user name who modified the version click **Advanced Search Options**. [Table 10-4](#) provides specific instructions on each criteria. You can use any or all of the criteria. If a criteria is left blank, it is not used in the search.

Table 10-4 Advanced Search Options for Searching the Version History of a Content Item

Search Criteria	Usage
Modified By	Searches for content versions that were modified by user name.
Workflow Status	Searches for content versions by workflow status.
Checked in	Searches for content versions by date.

- c. After defining your search criteria, click **Search**.

Publishing a Different Version of Content

In some cases you may want to change the published content within your portal. You can do this by checking out a previous version of a content item and updating it to Published status. To do this, check out the version of content you want to publish and check it back in with the Published status.

Note: This example assumes you are using the default content workflow. Your content workflow may have different status names. See your content administrator if you have questions.

To publish under a different version:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Select **Repository View**.
4. In the resources tree, click the content item you want to publish.
5. Click the **Version History** tab.
6. In Browse Versions section, click the **Check Out** icon for the version you want to publish. See [Figure 10-8](#).

Figure 10-8 Browse Versions

Version	Status	Modified On	View	Check Out
4 *Currently Published	Published	January 2, 2008		
3	Published	January 2, 2008		
2	Ready for Approval	January 2, 2008		
1	Draft	January 2, 2008		

7. Optionally, make any changes to the content.
8. Click Check In.
9. In Check In Content dialog, change the status of the content item to **Published** and click **Check In**. The earlier version is updated to the latest version and published.

Changing the Workflow of Content

Content created in a library services-enabled BEA repository uses the default content workflow, unless a customized workflow has been implemented. The default content workflow includes the following statuses:

- Draft
- Ready for Review
- Rejected
- Published
- Retired

Note: Depending on how your administrator has set up security for your content repository, you may not have access to all workflow statuses.

When using a library services-enabled BEA repository, you can change the content workflow that is associated with a content item. For example, if all of your content uses the default workflow, but you would like a particular content item to follow a different workflow, you can change the workflow associated with that content. For information about creating content workflows, see [“Creating Content Workflows” on page 5-4](#).

Note: It is possible to select a content workflow that is not compatible with the current workflow, see [“Using Content Workflows in Your BEA Repository” on page 5-1](#) for more details or ask your content administrator.

To change the workflow:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Content**.
3. Using either the **Repository View** or the **Workspace View > Assigned Items** folder, click the content you want to edit in the resources tree.
4. In the Versioning & Workflow section of the Summary page, click **Check Out**.
5. If using the Workspace View, select the content you want to edit in the Checked-Out Items folder.
6. In the Summary tab, click **Versioning & Workflow**.

Adding Content to a BEA Repository

7. In the Update Workflow dialog, select a content workflow from the New Workflow drop-down list and click **Update**.
8. If finished modifying your content, click **Check In**.

Using Display Templates

Display templates allow you to create standard views for portal content that you can use within your JSPs using display template JSP tags. For example, if your portal displays a list of books, you can create display templates that determine how the list of books is presented. Or you can create display templates that configure how an employee directory of pictures are presented.

Content management features such as the Content Presenter portlet and syndication feeds use display templates to present content to users. You can customize these templates or add new ones to extend these features.

This chapter includes the following topics:

- [Using Display Template JSP Tags](#)
- [Customizing the Content Presenter Configuration Wizard](#)
- [Using the Content Display Template Wizard](#)

Using Display Template JSP Tags

You use display template JSP tags to call display templates within your portal. Display templates are JSPs that you create that dictate how content is presented. After creating a display template, you register it with a `wlp-template-config.xml` file. After a display template (JSP) has been created and registered correctly, you can use it within a JSP using display template JSP tags.

WebLogic Portal provides two types of display template tags:

- `<dt:displaycmtemplate>` Displays the content types from your content management repository.
- `<dt:displaytemplate>` Displays other portal resources, such as HTML or JSPs.

For more information about display template JSP tags, see the [Commerce and Interaction Management JSP Tag Javadoc](#).

Before using a display template JSP tag, you must do the following:

- Create a display template
- Register the template in a `wlp-template-config.xml` file

Creating Display Templates

Display templates are JSP pages that you use in conjunction with display template JSP tags to display associated resources. These JSPs can incorporate any JSP functionality you require. You can use display templates to display repository content or other portal content, such as JSPs.

Tip: As a best practice, it is helpful to store all display templates in the same directory in your web application.

You can create two types of display templates in your portal; each are registered differently and associated with the respective JSP tag.

- Use content display templates to display repository content. These templates are registered for use with respective content types. You can use content display templates with the `<dt:displaycmtemplate>` tag.
- Use display templates are used to display any other portal resource. They are used with the `<dt:displaytemplate>` tag.

Within the `wlp-template-config.xml` file, there are two “root” elements used to register display templates:

- `<content-repository>` These templates are called content display templates.
- `<template-group>` These templates are called display templates.

Using Views

You can create multiple display template JSPs for the same resource. Typically, you create multiple views for displaying the same content type with a content display template. For example, you can create a JSP that displays a thumbnail image and another JSP that displays a full image. You can also assign a default view to use for a resource. You then register each template as a view within the `wlp-template-config.xml` file. Each template you register must have at least one view. [Listing 11-3](#) gives an example of a content display template tag that uses a particular view.

Using Content Display Templates within Display Templates

It is possible to re-use templates within other templates by incorporating a template JSP tag within your JSP. By re-using display templates in this way, you can reduce the amount of code you need to maintain. For example, Content Presenter templates use display templates to display content within the Content Presenter portlet and re-use the same content display template whenever possible. The content display template contains the code needed to retrieve the content node. In this case, if you want to change what content is retrieved (such as including a new property), you need only update the content display template.

Tip: A content display template is associated with a particular content type.

Creating a `wlp-template-config.xml` File

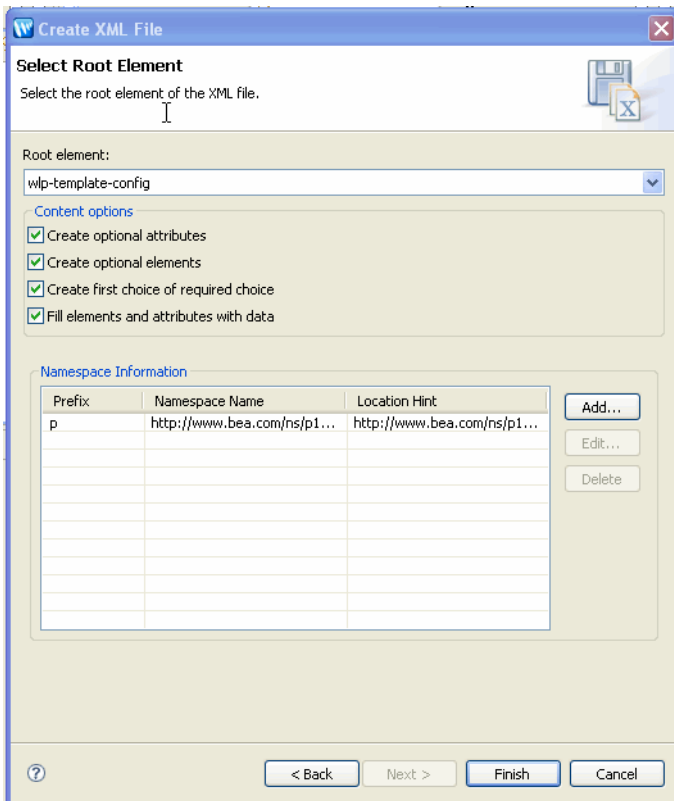
After creating a display template, you must create a configuration file that associates the template (JSP page) that you created with the respective resource. For example, if you have created a view to use with a content type, you must register that relationship using a `wlp-template-config.xml` file.

To create a new `wlp-template-config.xml` file using Workspace Studio:

1. In the Project Explorer, right-click your `//<WebProject>/WEB-INF` folder and select **New > Other**.
2. In the Select a Wizard dialog, navigate to the **XML > XML** selection and click **Next**.
3. In the Create XML File dialog, select **Create XML file from an XML schema file** and click **Next**.
4. In the XML File Name dialog, type `wlp-template-config.xml` for the name of the new file and click **Next**. Be sure the file is located in the `WEB-INF` directory of your web project.

5. In the Select XML Schema File dialog, mark **Select from XML Catalog**.
6. Select `http://www.bea.com/ns/p13n/90/wlp-template-config.xml` and click **Next**.
7. In the Select Root Element dialog, mark the Content Option check boxes for **Create optional attributes** and **Create optional elements**; see [Figure 11-1](#).

Figure 11-1 Selected Content Options in the Create XML File dialog



8. Click **Finish**.

9. After creating your `wlp-template-config.xml` file, right-click it and select Open With > XML Editor (or the editor of your choice) to edit the file. Use the following examples to assist you when editing the file.
 - [Listing 11-1](#) provides an commented outline of the `wlp-template-config.xml` schema.
 - [Listing 11-2](#) provides an example of a `wlp-template-config.xml` file.

Listing 11-1 `wlp-template-config.xml`

```

<wlp-template-config>
  <!-- Describes the content repository's types and templates -->
  <content-repository>

    <!-- Name of repository. (Required) -->
    <name> </name>

    <!-- Optionally, used to define a namespace for this set of templates.-->
    <content-name-space>

      <!-- Description for developer use, not shown to end user (Optional) -->
      <description> </description>

      <!-- If only the repository that matches this template is used to render
      content (Optional) -->
      <default-template-uri> </default-template-uri>

      <!-- Describes the content type -->
      <content-resource>

        <!-- Description not shown to end user (Optional) -->
        <description> </description>

        <!-- Name of content type.(Required) -->
        <name> </name>

        <!-- If only the content type matches, this template is used to
        render content. Contains the URI to the JSP location. (Optional) -->
        <default-template-uri> </default-template-uri>

        <!-- Describes the view (Optional) -->
        <view>

          <!-- Name of the view (Required) -->
          <name> </name>

          <!-- Description not shown to end user (Optional) -->
          <description> </description>

```

Using Display Templates

```
        <!-- If the view name that matches, this template is used to
        render content. Contains the URI to the JSP location.
        (Required) -->
        <uri> </uri>
    </view>
</content-resource>
</content-name-space>
</content-repository>

<!-- Describes templates that are not linked to a content repository --!>
<template-group>
    <name></name>

    <!-- Optional. Allows you to namespace your templates for organizational
    purposes. If creating templates for the Content Presenter, use
    wlp-content-presenter-multiple or wlp-content-presenter-single.--!>
    <template-name-space>
        <name></name>
        <template>
            <description> </description>

            <!-- (Required)Name of template. This name displays as the Template
            Category name within the Content Presenter portlet. --!>
            <name></name>

            <!-- Enter a default template URI. If your template definition
            contains no view elements, then this element is mandatory or the
            template will not work with the Content Presenter. --!>
            <default-template-uri> </default-template-uri>

            <!-- Optionally, specific a preview icon for this template --!>
            <default-template-preview-icon-uri>
            </default-template-preview-icon-uri>
            <view>
                <name> </name>
                <description> </description>
                <uri> </uri>

                <!--Optionally, specify a preview icon for this view --!>
                <preview-icon-url></preview-icon-url>
            </view>
        </template>
    </template-name-space>
</template-group>
</wlp-template-config>
```

[Listing 11-2](#) provides an example of a valid `wlp-template-config.xml` file.

Listing 11-2 Sample wlp-template-config.xml Document

```

<wlp-template-config>
  <content-repository>
    <name>MyRepo</name>
    <default-template-uri>/MyDefault.jsp </default-template-uri>
    <content-resource>
      <name>thumbnail</name>
      <view>
        <name>small</name>
        <uri>/smallview.jsp</uri>
      </view>
    </content-resource>
    <content-resource>
      <name>Product</name>
      <default-template-uri>/MyProductDisplay.jsp</default-template-uri>
      <view>
        <name>small</name>
        <uri>/product/smallview.jsp</uri>
      </view>
    </content-resource>
    <content-resource>
      <name>Camera </name>
      <default-template-uri>
        /MyCameraProductDisplay.jsp
      </default-template-uri>
      <view>
        <name>small</name>
        <uri>/product/camera/smallview.jsp</uri>
      </view>
    </content-resource>
  </content-repository>
</wlp-template-config>

```

Using the `<dt:displaycmtemplate>` Within a JSP

The `<dt:displaycmtemplate>` is used to display content from your content repository. Content display templates are associated with content types and can have multiple views. See the [Commerce and Interaction Management JSP Tag Javadoc](#) for more information about this tag and its attributes.

<dt:displaycmtemplate> Example

This example displays a content type of DigitalCamera which extends Camera which extends Product. See “<dt:displaycmtemplate> Example Using Type Inheritance” on page 11-10 for an example of how the tag takes advantage of type inheritance. It consists of three parts:

- [Listing 11-3](#) – Provides an example JSP tag usage that uses a defined view
- [Listing 11-4](#) – Provides an example usage that uses the default view
- [Listing 11-5](#) – Provides the corresponding configuration file for these examples

Tip: You can define a default view within your configuration file that is used for all content resources that do not have otherwise defined display templates. You do this by defining a content resource with a * as a value. For example, <name>*<name>. See [Listing 11-5](#) for an example. You can also use a * as <name> for the content-repository element.

[Listing 11-3](#) shows an example JSP tag usage that uses a defined view and how to specify the node with <cm:getNode>.

Listing 11-3 <dt:displaycmtemplate> Tag Usage

```
<cm:getNode path="/BEA Repository/sampleFolder/sampleNode" id="aNode"/>
<dt:displaycmtemplate repositoryName="MyRepo" resourceName="DigitalCamera"
view="small"/>
```

[Listing 11-4](#) shows an example usage that uses the default view.

Listing 11-4 <dt:displaycmtemplate> Tag Usage

```
<dt:displaycmtemplate repositoryName="MyRepo" resourceName="Camera"
view="small"/>
```

Listing 11-5 provides the corresponding configuration file.

Listing 11-5 Corresponding wlp-template-config.xml

```

<wlp-template-config>
  <content-repository>
    <name>MyRepo</name>
    <default-template-uri>/MyDefault.jsp </default-template-uri>
    <content-resource>

      <!-- Using an * in the name element defines a default view for content
      resources that do not have registered templates.--!>
      <name>*</name>

      <view>
        <name>small</name>
        <uri>/smallview.jsp</uri>
      </view>
    </content-resource>
    <content-resource>
      <name>Product</name>
      <default-template-uri>/MyProductDisplay.jsp</default-template-uri>
      <view>
        <name>small</name>
        <uri>/product/smallview.jsp</uri>
      </view>
    </content-resource>
    <content-resource>
      <name>Camera</name>
      <default-template-uri>
        /MyCameraProductDisplay.jsp
      </default-template-uri>
      <view>
        <name>small</name>
        <uri>/product/camera/smallview.jsp</uri>
      </view>
    </content-resource>
  </content-repository>
</wlp-template-config>

<!-- With this configuration the uri of "product/camera/smallview.jsp" will be
found. This is because the Digital Camera content type inherits from the Camera
content type and the camera <content-resource> has a small view. -->

```

<dt:displaycmtemplate> Example Using Type Inheritance

This example displays a content type of DigitalCamera which extends Camera which extends Product. Notice that the tag leverages type inheritance when searching for which view to use. For more information about type inheritance, see [“Using Content Type Inheritance” on page 6-3](#).

[Listing 11-6](#) shows the syntax of the JSP tag.

Listing 11-6 <dt:displaycmtemplate> Tag Usage

```
<dt:displaycmtemplate repositoryName="MyRepo" resourceName="DigitalCamera"
view="full"/>
```

[Listing 11-7](#) provides the corresponding configuration file.

Listing 11-7 Corresponding wlp-template-config.xml

```
<wlp-display-template>
  <content-repository>
    <name>MyRepo</name>
    <default-template-uri>/MyDefault.jsp </default-template-uri>
    <content-resource>
      <name>Product</name>
      <default-template-uri>/MyProductDisplay.jsp</default-template-uri>
      <view>
        <name>small</name>
        <uri>/product/smallview.jsp</uri>
      </view>
      <view>
        <name>full</name>
        <uri>/product/fullview.jsp</uri>
      </view>
    </content-resource>
  </content-repository>
  <content-resource>
    <name>Camera</name>
    <default-template-uri>
      /MyCameraProductDisplay.jsp
    </default-template-uri>
    <view>
      <name>small</name>
      <uri>/product/camera/smallview.jsp</uri>
    </view>
  </content-resource>
</wlp-display-template>
```

```

    </content-resource>
  </content-repository>
</wlp-display-template>

```

```

<!-- With this configuration the uri of "product/fullview.jsp" will be found.
This is the JSP tag takes advantage of type inheritance and knows that camera
inherits from product. -->

```

Using the `<dt:displaytemplate>` Tag Within a JSP

Use the `<dt:displaytemplate>` tag to display any non-content management portal resources. For more information about this tag and its attribute, see the [Commerce and Interaction Management JSP Tag Javadoc](#).

Customizing the Content Presenter Configuration Wizard

The Content Presenter portlet provides a quick way to show customized content within your portal. With the Content Presenter portlet, any portal user with sufficient rights can change which content is displayed in the portlet. For more information about the Content Presenter portlet and configuration instructions, see [Using the Content Presenter Portlet](#) in the [Portlet Development Guide](#).

You can create custom display templates for the Content Presenter portlet. To use custom display templates within Content Presenter, you must register them specifically for Content Presenter within the `wlp-template-config.xml` file.

For example, you can create one template to control how the details of a single press release are displayed, and you can create another template to control how a list of multiple press releases are displayed. You allow portlet users to choose different views such as one that displays a summary of the press release, the press release's title only, or an internal view of the press release.

For general information about creating and configuring display templates, see [“Using Display Template JSP Tags” on page 11-1](#).

Note: When the user does not have authorization to view the content displayed in the Content Presenter portlet, the default behavior is to ignore the error and display an empty portlet. As the template designer, if you want to display a message instead, you must create the required logic in the template for handling the Authorization Exception.

Using Templates with Content Presenter

Content display templates cannot be used directly in Content Presenter.

However, you can use content display templates within your Content Presenter templates. By calling content display templates within a display template JSP, you can re-use code for different presentation styles.

Tip: The Content Presenter portlet uses generic display templates within its implementation that are called by the `<dt:displaytemplate>` JSP tag. Content display templates are called with the `<dt:displaycmtemplate>` JSP tag; therefore they cannot be used directly by the Content Presenter portlet.

Using content display templates is optional and can make your template configuration more complicated. However, when you re-use content display templates within your Content Presenter templates, you can take advantage of content type inheritance, see [“<dt:displaycmtemplate> Example Using Type Inheritance” on page 11-10](#).

Creating New Display Templates for the Content Presenter Wizard

This section contains an example of adding and configuring two new templates that can be used by the Content Presenter portlet.

First, you create a content display template that retrieves a press release content node and its title. Next you call that template, using the `<dt:displaycmtemplate>` tag, from two other display templates that each list the press release titles with different types of bullets. Finally, you register each template for use in the Content Presenter portlet by creating a `wlp-template-config.xml` file.

Creating a Content Display Template to Reuse

Optionally, you can use content display templates within your display templates to centralize code that retrieves content from your repository. In this example, you create a content display template that retrieves the title of a press release. You can later re-use this content display template in a display template registered for the Content Presenter. This way you avoid duplicating the code in each display template that displays the press release title.

[Listing 11-8](#) shows a content display template JSP file that will be re-used in a display template. Notice the use of the `<cm:getProperty>` that is used to retrieve content.

Listing 11-8 A Sample Content Display Template Called pressReleaseTitleOnly.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@ taglib uri="http://www.bea.com/servers/portal/tags/content"
prefix="cm"%>
<cm:getProperty name="title"/>

```

Each template you create must be registered in the `wlp-template-config.xml`. When registering content display templates created for re-use in display templates designed for the Content Presenter portlet, you should include the `<content-name-space>` element and give it a `wlp-content-presenter` value, see [Listing 11-9](#).

For more information about how content display templates are registered, see [“Register Your Templates for Content Presenter Portlet” on page 11-15](#).

Note: If you want to overwrite an existing content display template that is being used, you can use the Content Display Template Wizard, see [“Using the Content Display Template Wizard” on page 11-21](#).

Listing 11-9 Example wlp-template-config.xml File for a Content Display Template to be Re-used in a Content Presenter Display Template

```

...
<content-repository>
  <name>BEA Repository</name>
  <content-name-space>
    <name>wlp-content-presenter-multiple</name>
    <content-resource>
      <name>pressRelease</name>
      <default-template-uri>
        /templates/cm/pressReleaseDefault.jsp
      </default-template-uri>
      <view>
        <name>Press Release Title Only</name>
        <description>
          This retrieves on the title property of a pressRelease.
        </description>
        <uri>/templates/cm/pressReleaseTitleOnly.jsp</uri>
      </view>
    </content-resource>
  </content-name-space>

```

```
</content-repository>  
...
```

Creating Display Templates for the Content Presenter Portlet

You can design your display templates with all the flexibility allowed by JSPs. The example in this section provides two different display templates (JSPs) that each prepend different bullets in front of a list of press release titles.

Use the `<dt:displaycmtemplate>` tag in these JSPs to call the `pressReleaseTitleOnly` content display template, as described in [“Creating a Content Display Template to Reuse” on page 11-12](#). This helps minimize template code, enforces common presentation patterns, and allow you to take advantage of content type inheritance. For more information, see [“<dt:displaycmtemplate> Example Using Type Inheritance” on page 11-10](#).

[Listing 11-10](#) provides an example of template that uses a text bullet to prepend the list of press release titles.

Listing 11-10 A Template View Called `pressReleaseListBulletText.jsp` (using a text bullet)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>  
<%@ taglib uri="http://www.bea.com/servers/portal/tags/dt" prefix="dt"%>  
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  
<%@ taglib tagdir="/WEB-INF/tags/wlpapps/contentui/templates"  
    prefix="templateTags"%>  
<templateTags:getTemplateNodes var="nodes"/>  
<c:forEach items="${nodes}" var="node">  
    &bull;  
    <dt:displaycmtemplate  
        resourceName="${node.objectClass.name}"  
        view="Press Release Title Only"  
        ** This node is passed to the content display template.**  
        node="${node}"/>  
    <br/>  
</c:forEach>
```

[Listing 11-11](#) provides an example of a template that uses a .gif image.

Listing 11-11 A Template View Called `pressReleaseListBulletImage.jsp` (using a *.gif)

```
<%@ page language=" java" contentType="text/html; charset=UTF-8"%>
<%@ taglib uri="http://www.bea.com/servers/portal/tags/dt" prefix="dt"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib tagdir="/WEB-INF/tags/wlpapps/contentui/templates
"prefix="templateTags"%>
<templateTags:getTemplateNodes var="nodes"/>
<c:forEach items="${nodes}" var="node">
    
    <dt:displaycmtemplate
        resourceName="${node.objectClass.name}"
        view="Press Release Title Only"
        node="${node}"/>
    <br/>
</c:forEach>
```

Register Your Templates for Content Presenter Portlet

You use a `wlp-template-config.xml` file to register templates for the Content Presenter Portlet.

When you register templates for the Content Presenter portlet, be sure to include the required elements to ensure the templates are accessible. The values for some of these elements are also shown to users when they use the Content Presenter Configuration Wizard wizard. For example, you must use appropriate namespaces for both the content display templates and display templates to ensure that the templates can be used in Content Presenter.

[Listing 11-12](#) provides a blank example of the configuration elements that are re-used in the Content Presenter Configuration Wizard. See [Table 11-1](#) for detailed information about these required elements.

Listing 11-12 Blank Example of wlp-template-config.xml including Elements used in Content Presenter Wizard

```
<template-group>
  <name>My Custom Template Category</name>
  <template-name-space>
    <name>wlp-content-presenter-multiple</name>
    <view>
...
  </template-name-space>
</template-group>

OR

<template-group>
  <name>My Custom Template Category</name>
  <template-name-space>
    <name>wlp-content-presenter-single</name>
    <view>
...
  </template-name-space>
</template-group>
```

[Table 11-1](#) lists the configuration elements specific to configuring a display template for use in the Content Presenter portlet. [Listing 11-13](#) provides an example of these elements as they should appear in the `wlp-template-config.xml` file when configuring the new templates used in this example.

Note: For a complete example of the `wlp-template-config.xml`, see [Listing 11-1](#), “`wlp-template-config.xml`,” on page 11-5.

Table 11-1 Configuration Elements Specific to Content Presenter Templates

wlp-template-config.xml Element	What it does:
<template-group>	<p>Defines a group of templates for the Content Presenter portlet.</p> <p>Template groups are represented in the Content Presenter Configuration Wizard portlet as “Template Categories”.</p>
<template-name-space> (child element of <template-group>)	<p>Defines the namespace of a template or category (as seen in the Content Presenter portlet). The namespace is required and allows you to differentiate whether the templates are available to present multiple or single content items.</p> <p>In the Content Presenter Configuration Wizard, the value of this namespace determines if the template is available for multiple or single content items.</p>
<name> (child element of <template-name-space>)	<p>The value of this element determines if the template or view displays as a selection for a multiple or single item template within the Content Presenter Configuration Wizard.</p> <p>This element must have one of the following values:</p> <ul style="list-style-type: none"> <li data-bbox="673 1025 1228 1225">• <code>wlp-content-presenter-multiple</code> Any templates within the <code>wlp-content-presenter-multiple</code> namespace appear when the user selects the option to show multiple content items in the wizard. Therefore, all templates placed under this namespace should be written to handle multiple content items. <li data-bbox="673 1242 1228 1347">• <code>wlp-content-presenter-single</code> The <code>wlp-content-presenter-single</code> namespace is designed for templates that handle a single content item.
<default-template-uri>	<p>Defines a default template JSP to be used for this template category. If you use a template view within this category, you can omit this element.</p> <p>If you do not have any template views for this namespace, the <code><default-template-uri></code> element is required.</p>

Tip: Defining multiple views within a template category can help organize your content and make it easier to use the Content Presenter Configuration Wizard. For example, you create a template category called *Press Releases* that has a `<default-template-uri>`. You might need several ways to view Press Releases, such as a *Press Release Summary*, *Press Release Title Only*, *Press Release Public View*, and *Press Release Internal View*. In this case, creating multiple template views under your *Press Releases* template category is a good solution. For more information about views, see “Using Views” on page 11-3.

Listing 11-13 shows a sample `wlp-template-config.xml` file that references the content display template created earlier (see Listing 11-8) and both display templates that are used directly by the Content Presenter portlet (see Listing 11-10 and Listing 11-11).

Note: This example also includes a content display template that cannot be used within the Content Presenter. For more information about using display templates, see “Using Display Template JSP Tags” on page 11-1.

Listing 11-13 A `wlp-template-config.xml` File That Includes the New Templates

```
<?xml version="1.0" encoding="UTF-8"?>
<wlp-template-config
xmlns="http://www.bea.com/ns/pl3n/90/wlp-template-config">
<content-repository>
  <name>BEA Repository</name>
  <!--The following bold section is an example of a content display
template that not been namespaced for Content Presenter use. It can
still be used within a Content Presenter template.-->
  <content-resource>
    <name>Product</name>
    <default-template-uri>
      /MyProductDisplay.jsp
    </default-template-uri>
    <view>
      <name>small</name>
      <uri>/product/smallview.jsp</uri>
    </view>
  </content-resource>
</content-repository>
</wlp-template-config>
```

```

<view>
  <name>full</name>
  <uri>/product/fullview.jsp</uri>
</view>
</content-resource>

```

<!-- The following bold section is an example of the best practice of using a namespace for a content display template that is re-used within a template for the Content Presenter portlet. -->

```

<!-- This name element creates a namespace for this template -->
<content-name-space>
  <name>wlp-content-presenter-multiple</name>
  <content-resource>
    <name>pressRelease</name>
    <default-template-uri>
      /templates/cm/pressReleaseDefault.jsp
    </default-template-uri>
    <view>
      <name>Press Release Title Only</name>
      <description>
        This retrieves on the title property of a pressRelease.
      </description>
      <uri>/templates/cm/pressReleaseTitleOnly.jsp</uri>
    </view>
  </content-resource>
</content-name-space>
</content-repository>

```

<!--The following bold section is an example of a display template that is used directly by the Content Presenter template. It includes a template group, a template namespace, and also uses the view element to list different templates associated with this category. Optionally, it includes a preview icon for both the default template and each view to aid users in selecting the correct template in the wizard. -->

```

<template-group>
  <name>Custom Template Category</name>
  <template-name-space>
    <name>wlp-content-presenter-multiple</name>

```

```
<template>
  <description>
    This is the default template for listing Press Releases.
  </description>
  <name>Press Release List Template</name>
  <default-template-uri>
    /templates/pressReleaseList.jsp
  </default-template-uri>
  <default-template-preview-icon-uri>
    /templates/images/pressReleaseList.gif
  </default-template-preview-icon-uri>
  <view>
    <name>Press Release Bulleted List - Text</name>
    <description>
      This lists Press Releases with text bullets.
    </description>
    <uri>/templates/pressReleaseListBulletText.jsp</uri>
    <preview-icon-uri>
      /templates/images/pressReleaseListBulletText.gif
    </preview-icon-uri>
  </view>
  <view>
    <name>Press Release Bulleted List - Image</name>
    <description>
      This lists Press Releases with imagebullets.
    </description>
    <uri>/templates/pressReleaseListBulletImage.jsp</uri>
    <preview-icon-uri>
      /templates/images/pressReleaseListBulletImage.gif
    </preview-icon-uri>
  </view>
</template>
</template-name-space>
</template-group>
</wlp-template-config>
```

Using the Content Display Template Wizard

In Workspace Studio, you can use the Content Display Template Wizard to create new content display templates. The Content Display Template Wizard generates the appropriate JSP and generates a file that registers the template according to the repository and content type you choose.

After creating the content display template, you can customize it (using the JSP editor in Workspace Studio, for example). As long you do not change the name of the display template, it remains registered.

The Content Display Template Wizard in Workspace Studio uses the `<dt:displaycmtemplate>` tag to create content display templates.

Tip: Templates that you create with the Content Display Template Wizard do not automatically appear in the Content Presenter Configuration Wizard, because the configuration wizard only shows templates that were created with the `<dt:displaytemplate>` tag. If you created a template with the Content Display Template Wizard, you can make the template appear in the Content Presenter Configuration Wizard by calling the template you created in Workspace Studio (`<dt:displaycmtemplate>`) from a regular display template (`<dt:displaytemplate>`) that you create in the Content Presenter Configuration Wizard.

Updating Content Presenter Display Templates

You can update content display templates that are referenced in the Content Presenter display templates using the Content Display Template Wizard. If you select a previously registered namespace and view, the content display template you create can be called by existing Content Presenter display templates. For more information about configuring Content Presenter display templates, see [“Creating New Display Templates for the Content Presenter Wizard” on page 11-12.](#)

If you want to change or update the content properties that are retrieved for a Content Presenter display template, you can overwrite an existing view with the Content Display Template Wizard. For example, the `wlp-default-list` view (WLP Default List) lists the primary property of a content item. If you want to update the `wlp-default-list` view to list additional properties, you can use the Content Display Template Wizard to create a new JSP (that uses the same view name of `wlp-default-list`) that includes only the content type properties you select.

Creating a Content Display Template

To create a content display template in WorkSpace Studio:

1. Switch the portal perspective if you are not already using it. Select **Window > Open Perspective > Other > Portal**.
2. Choose **File > New > Other**.
3. In the Select a Wizard window, expand the **WebLogic Portal** directory, select **Content Display Template**, and click **Next**.
4. In the Select Project window, highlight your portal web project and click **Next**.
5. In the Content Properties window, choose the content properties you want to display with your template.

Note: When you use the wizard to create a content display template, you can either access the content types and properties on your server or enter them manually if you do not have access to the server.

If you are running a server:

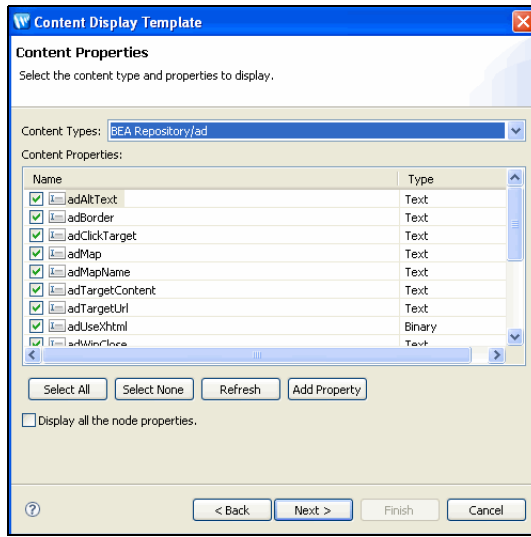
- Select a content type from the Content Types field. The drop-down menu lists all available content types in your repository.
- Choose which content properties to display in the template by selecting the corresponding check boxes. You can also click **Select All**. See [Figure 11-2](#).

Note: The Content Display Template Wizard does not support nested or linked content properties. If you want to use these properties in your template, you need to manually edit the content display template JSP.

- Click **Next**.

If you are not running a server:

- Click **Add Property** to add each content type property to the wizard that you want to display. Enter the name and data type of the property and click **OK**.
- You can also select the **Display all node properties** check box to retrieve all content type properties at run time. You can then manually edit the display template to update or change the way the properties are retrieved (remove properties, change sort order and so on).
- When finished adding content type properties, click **Next** to continue to the Template Information page. See [Figure 11-2](#).

Figure 11-2 Choose which Content Properties to Display in the Template

6. In the Template Information page, use the drop-down menu to select a Repository Name, Namespace, and Content Type.
 - If you are updating a content display template used by the Content Presenter Configuration Wizard, your namespace must be either **wlp-content-presenter-multiple** or **wlp-content-presenter-single**. See [“Register Your Templates for Content Presenter Portlet”](#) on page 11-15 and see the [Tip](#): at the end of this section.
 - You can also use an asterisk (*) for the content repository or content type, and the template matches any content repository or content type.
7. In the View Information window, enter the template view name and information.
 - a. In the View Name field, select an existing view (populated from the current display template configurations) to overwrite the existing configuration. You can also create a new view by typing a new name.

[Table 11-2](#) lists the content display template views available in WorkSpace Studio. These template views are not automatically visible in the Content Presenter Configuration Wizard, unless you perform additional steps (see the [TIP](#) at the end of this section).

Table 11-2 Content Presenter Content Display Template Views

View	What it does:
WLP Default Syndication View	Lists all properties of the content item in a syndication feed. For more information, see “Creating the Syndicated Feed JSPs” on page 12-14.
WLP Default Details	Lists all properties of the content item.
WLP Default Single Property	Lists a single node and a single property.
WLP Default List	Lists the primary property of a content item.
WLP Content Presenter Wizard Item Details	Lists all properties of the content item within the wizard

Note: You must have added the Content Presenter facet to your web project in order to access the Content Presenter template views. For more information about the Content Presenter portlet, see the [Portlet Guide](#).

- b. You can choose to enter a description of your view in the View Description field.
 - c. You can also choose to associate an icon with this view. When you associate an icon with a view, the icon is used to assist users in choosing the correct template view in the Content Presenter Configuration Wizard.
 - d. Click **Next**.
8. In the View JSP page, save the new template to a location within your web project.
 - a. Enter the JSP in the File Name field, including the .jsp extension.
 - b. If the file already exists and the **Overwrite existing resource?** check box is not selected, the file is not modified.
 - c. Select the **Open the JSP file when wizard is finished?** check box to immediately view the JSP in the JSP editor.
 - d. Select the **Use JSTL tags in template?** check box if your application uses the JavaServer Pages Standard Tag Library (JSTL).
 9. Click **Finish** to view the JSP file in the JSP Editor in WorkSpace Studio.

The template and view configuration information is stored in the `wlp-template-config.xml` file and the new JSP is created.

Tip: Templates that you create with the Content Display Template Wizard in WorkSpace Studio do not automatically appear in the Content Presenter Configuration Wizard, because the configuration wizard only shows templates that were created with the `<dt:displaytemplate>` tag. If you created a template with WorkSpace Studio's Content Display Template Wizard, you can make the template appear in the Content Presenter Configuration Wizard by calling the template you created in WorkSpace Studio (`<dt:displaycmtemplate>`) from a regular display template (`<dt:displaytemplate>`) that you create in the Content Presenter Configuration Wizard.

Using Display Templates

Using Syndicated Feeds

A syndicated feed, also called a web feed, provides users with frequently updated content. Syndicated feeds allow you to publish content to a third-party feed reader. For example, content publishers can configure a feed reader on their local machine that reads a syndicated feed that includes all content that is ready to publish. By subscribing to a syndicated feed, using a feed reader or aggregator, a user stays up to date on content changes.

WebLogic Portal provides a Syndication Producer Servlet and several preconfigured syndicated feeds that you can modify. You can also create your own custom feeds. The preconfigured syndicated feeds are compatible with RSS feed technology and consist of URL formats that can be read by most RSS readers. They conform to the RSS 2.0 Specification.

Note: Internet Explorer 6 does not support syndicated feeds.

This chapter includes the following sections:

- [Using and Modifying the Preconfigured Syndicated Feeds](#)
- [Creating Custom Syndicated Feeds](#)
- [Securing Syndicated Feeds](#)

Using and Modifying the Preconfigured Syndicated Feeds

The preconfigured syndicated feeds include basic examples of using display template JSPs to select and display content. These JSPs display general metadata, such when the content was created, who created it, and the name of the content node. They are generic display templates that show the information in a simple format.

Tip: Use these basic preconfigured syndicated feeds as a starting place for developing your own custom syndicated feeds.

URL Format for Syndicated Feeds

The URL format for syndicated feeds is shown in [Listing 12-1](#).

Listing 12-1 URL Format for Syndicated Feeds

```
http://<Administration Console Host Machine Name>: <Administration Console Port  
default is 7001>/<WebApp>/SyndicationProducer?feedName=<feedName>  
&syndicationStyleName=<styleName>
```

[Table 12-1](#) lists the three preconfigured syndicated feed URLs provided with WebLogic Portal.

Table 12-1 Pre-Configured Syndicated Feeds

Feed Name	Description	Syndicated Feed URL
Latest Content	Shows the most recent files added to the repository.	<code>http://<hostname>:7001/<WebApp>/SyndicationProducer?feedName=LatestContent&syndicationStyleName=rss</code>
Need to Approve	Shows the documents that need approval before publishing.	<code>http://<hostname>:7001/<WebApp>/SyndicationProducer?feedName=NeedToApprove&syndicationStyleName=rss</code>
Directory Contents	Shows the files in a particular directory with the latest files at the top of the directory.	<code>http://<hostname>:7001/<WebApp>/SyndicationProducer?feedName=DirectoryContents&syndicationStyleName=rss</code>

Changing the Search Results Using a URL

If you want to change the search results for a particular feed, you can modify its URL. Modifying the URL allows you to pass parameters into the Syndication Producer Servlet without modifying the syndication configuration file, which sets the parameters passed to the Syndication Producer Servlet. For example, to change the number of articles displayed in a syndicated feed and the reader type, you would add the `searchMaxResults` parameter between ampersands and change the value of the `syndicationStyleName` to `atom`. The resulting URL looks like:

```
http://www.bea.com:7001/myWebApp/SyndicationProducer?feedName=LatestContent&searchMaxResults=5&syndicationStyleName=atom
```

Note that the URL follows the format in [Listing 12-1](#). The available parameters are listed in [Table 12-2](#). Be sure to separate each parameter with an ampersand.

Note: Any parameters not set in the URL should be set in the syndication configuration (`wlp-syndication-config`) file. Otherwise, you may get undesirable results or failure. For example, if the `searchMaxResults` parameter is not set anywhere, the syndicated feed could display hundreds of entries.

For more information about the `wlp-syndication-config.xml` file, see [“Modifying the Syndication Configuration File” on page 12-9](#).

Modifying the URL is especially useful during development. You can use the URL to help determine and test the parameters that you want to pass into the Syndication Producer Servlet. This saves you time because changing syndication configuration file requires redeployment of the application. After establishing the correct parameters, you can edit the `wlp-syndication-config.xml` file and redeploy.

The parameters passed in through a URL have precedence over parameters passed in from the syndication configuration file. The Syndication Producer Servlet sets the priority in the following order:

1. URL query parameters
2. Syndication-feed settings in `wlp-syndication-config.xml`
3. Syndication-style settings in `wlp-syndication-config.xml`

Modifying the Preconfigured Syndicated Feeds

To modify the preconfigured syndicated feeds, you need to change the following files:

- Syndicated Feed JSPs – These files retrieve information from the repository and display the feed and content of each item.
- Syndication Configuration file – The `wlp-syndication-config.xml` file sets the parameters passed to the Syndication Producer Servlet and maps entries to the `wlp-template-config.xml` file.
- Display Template Configuration file – The `wlp-template-config.xml` file selects which JSPs used for the syndicated feed.

About the Syndicated Feed JSPs

WebLogic Portal contains four preconfigured JSPs. The first three JSPs generate the list of content items, and the fourth generates the content itself. See [Figure 12-1](#) and [Figure 12-2](#).

- `rss_header.jsp` – Header, which is the title of the syndicated feed, such as All Dev2Dev Articles.
- `rss_item.jsp` – List item, which is the individual listing for each article, specifically the items that the query finds. If the query finds six articles, this JSP is called six times.
- `rss_footer.jsp` – Footer, which closes the list.
- `rss_detailed_view.jsp` – Content or article, which is called when a user clicks a list item.

Figure 12-1 RSS Feed Example



[Figure 12-2](#) shows the article itself, which is displayed by the `rss_detailed_view.jsp`.

Figure 12-2 Article Displayed by rss_detailed_view.jsp

GET SERIOUS ABOUT SOA GOVERNANCE: A 5-STEP ACTION PLAN FOR ARCHITECTS

09/27/2007

Whether your organization's Service- Oriented Architecture (SOA) has 50 services in use by one customer, or 50 customers using one service, you need SOA governance in order to fully benefit from your SOA. Increased business and IT agility depends on SOA governance: the ability to quickly and continuously translate and transmit business strategy and requirements into the policies and standards that will guide the evolution of the SOA—and your enterprise.

Get Serious About SOA Governance: A 5-Step Action Plan for Architects, will help you get SOA governance right, and get what you expect out of your SOA.

- Download [Get Serious About SOA Governance: A 5-Step Action Plan for Architects](#) (registration required)

Modifying the Preconfigured Syndicated Feed JSPs

To modify the preconfigured syndicated feeds JSPs in Workspace Studio, from the Merged Projects View, copy the RSS JSP files from the `myPortalWebProject` folder to your web project, and then edit the files. For information on how to use Workspace Studio for this purpose, see “Using the Merged Projects View” in [Setting up Your Portal Development Environment](#) in the *Portal Development Guide*.

Retrieving Content

In content management, the two types of repositories are published and versioned. The nodes in a published repository provide only a single version of information; the user of a syndicated feed sees that information every time the content node is updated. The nodes in a versioned repository provide multiple versions that can be searched and managed. When you enable library services, your content is automatically versioned and a new copy of the content is saved whenever you check in content to the repository. Generally, you use versioned repositories in a workflow, where a manager approves the content before it is published. For more information about library services, see [“Using Versioning” on page 10-22](#).

Retrieving Information from a Published Repository

To call a node in a published repository, you set the search type in the syndication configuration file (`wlp-syndication-config.xml`) to `pubmeta`:

```
<search-type>pubmeta</search-type>
```

When the Syndication Producer Servlet calls the JSP to retrieve data from a content node, in the JSP's scriptlet, the request attribute of the node in the JSP's scriptlet is set to not null:

```
if(request.getAttribute("node") != null){
```

[Listing 12-4](#) shows the entire wlp-syndication-config.xml file and [Listing 12-2](#) shows an example JSP (rss_item.jsp).

Retrieving Information from a Versioned Repository

To call a versioned node, you set the search type in the syndication configuration file (wlp-syndication-config.xml) to vermeta:

```
<search-type>vermeta</search-type>
```

When the Syndication Producer Servlet calls the JSP to retrieve data from a versioned node, the request attribute version in the JSP's scriptlet is set to not null:

```
else if (request.getAttribute("version") != null){
```

[Listing 12-4](#) shows the entire wlp-syndication-config.xml file and [Listing 12-2](#) shows an example JSP (rss_item.jsp).

Example JSP

The following example JSP contains the logic to render the syndication feed. The Syndication Producer servlet uses the content query to retrieve the data from content management. The JSP displays the results returned from the content query.

Listing 12-2 Example of a Syndicated Feed JSP for an Item—rss_item.jsp

```
<%@page contentType="text/html;charset=UTF-8" language="java"%>
<%@ page import="com.bea.content.Node" %>
<%@ page import="com.bea.content.ContentContext" %>
<%@ page import="com.bea.content.virtual.version.Version" %>
<%@ page import="com.bea.content.federated.ContentManagerFactory"%>
<%@ page import="com.bea.content.federated.INodeManager"%>
<%

String path = request.getRequestURL().toString();
path = path.substring(0, path.indexOf(request.getContextPath()));
request.setAttribute("rootURL", path+"/"+ request.getContextPath());

    if(request.getAttribute("node") != null){
        Node node = (Node)request.getAttribute("node");
        if(node != null){
            String nodePath = node.getPath();
```

```

        request.setAttribute("parameters", request.getAttribute("rootURL") +
            "/ShowProperty?nodePath="+nodePath);
        request.setAttribute("path", request.getAttribute("rootURL") +
            "/ShowProperty?nodePath="+nodePath);
        request.setAttribute("title", node.getName());
    }
    else{
        request.setAttribute("parameters", "#");
        request.setAttribute("path", "Node could not be found from path");
        request.setAttribute("title", "Null Node Exception");
    }
}
else if (request.getAttribute("version") != null){
    Version version = (Version)request.getAttribute("version");
    if(version != null){
        INodeManager nodeMgr = ContentManagerFactory.getNodeManager();
        Node node = nodeMgr.getNodeByUUID(new ContentContext(),
            version.getNodeId());
        if(node != null){
            String nodePath = node.getPath();
            request.setAttribute("parameters", request.getAttribute("rootURL") +
                "/ShowProperty/"+nodePath+"||versionId="+version.getVersionName());
            request.setAttribute("path", request.getAttribute("rootURL") +
                "/ShowProperty/"+nodePath+"||versionId="+version.getVersionName());
            request.setAttribute("title", node.getName());
        }
        else{
            request.setAttribute("parameters", "#");
            request.setAttribute("path", "Node could not be found from path");
            request.setAttribute("title", "Null Node Exception");
        }
    }
    else{
        request.setAttribute("fullURL", "Version attribute was null");
        request.setAttribute("title", "Null Version");
        request.setAttribute("description", "Null Node");
    }
}
}
%>

<item>
    <title>${title}</title>
    <link>${fullURL}</link>
    <description>${description}</description>
</item>

```

Displaying the Contents of an Item

As previously mentioned, the JSP selects the data from the repository and displays it. The end of [Listing 12-2](#) contains the following HTML, which displays each item in a syndicated list:

```
<item>
  <title>${title}</title>
  <link>${fullURL}</link>
  <description>${description}</description>
</item>
```

The `rss_detailed_view.jsp` contains the HTML that displays the contents of the item in the list, that is, the article itself, as shown in [Listing 12-3](#).

Listing 12-3 Example Syndicated Feed JSP for an Article—`rss_detailed_view.jsp`.

```
...
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>View Details: ${title}</title>
<style>
  body {background-color:#FFFFFF; font-family:arial, sans-serif;
    font-size: 1em; color:#000000;}
  p td {font-family:arial, sans-serif; font-size: 100%; color:#000000;}
  h1 {font-size:120%; border-bottom: 1px solid #CCCCCC;}
  .notetext {font-size:80%; width: 50%; padding:10px; border: 1px solid
  #FFCC33; background-color: #FFFCC}
</style>
</head>
<body>
  <h1>View Details: ${title}</h1>
  <div class="notetext">In this view, you can display useful properties for
    the selected feed item, and format them with the display template. By
    default, we display a few system properties.</div>
  <p>Location in Repository: ${nodePath}</p>
  <p>Created on: ${createdDate}</p>
```

```

    <p>Created by: ${createdBy}</p>
    <p>Primary Property Details: <a href="${parameters}">View Binary</a></p>
</body>
</html>

```

Modifying the Syndication Configuration File

The syndication configuration file sets the parameters passed to the Syndication Producer Servlet and maps the Syndication Feed JSPs to a namespace, which allows you to change which display templates are used.

To modify the preconfigured syndicated feeds in WorkSpace Studio, from the Merged Projects View, copy the `WEB-INF\wlp-syndication-config.xml` file to your web project, and then edit the file. For information on how to use WorkSpace Studio for this purpose, see [Working with the Merged Projects View](#).

[Listing 12-4](#) shows the syndication configuration file (`wlp-syndication-config.xml`) for the preconfigured syndicated feeds.

Listing 12-4 Example of a Syndication Configuration File (`wlp-syndication-config.xml`)

```

<?xml version="1.0"?>
<wlp-syndication-config
xmlns="http://www.bea.com/ns/content/102/wlp-syndication-config">
  <syndication-feed>
    <name>LatestContent</name>
    <search-query>cm_nodeName like '*'</search-query>
    <search-type>pubmeta</search-type>
    <search-max-results>20</search-max-results>
    <search-sort-order>cm_createdDate desc</search-sort-order>
  </syndication-feed>
  <syndication-feed>
    <name>NeedToApprove</name>
    <search-query>cm_lifeCycleStatus = 3 && cm_assignedToUser =
      requestProperty('DefaultRequestPropertySet', 'Remote
      User')</search-query>
    <search-type>vermeta</search-type>
  </syndication-feed>
  <syndication-feed>
    <name>DirectoryContents</name>
    <search-query>cm_path like '/BEA Repository/*'</search-query>
    <search-type>pubmeta</search-type>

```

```
<search-max-results>20</search-max-results>
<search-sort-order>cm_createdDate desc</search-sort-order>
</syndication-feed>
<syndication-style>
  <name>rss</name>
  <name-space>bea-rss</name-space>
  <channel-group-name>ContentRepoRSS</channel-group-name>
  <channel-template>RSSChannel</channel-template>
  <channel-header-view>channelHeader</channel-header-view>
  <channel-footer-view>channelFooter</channel-footer-view>
</syndication-style>
<syndication-item-style>
  <name>rss_detailed_view</name>
  <name-space>bea-rss</name-space>
  <item-view>detailedView</item-view>
</syndication-item-style>
</wlp-syndication-config>
```

Search Types

There are two search types in the `<syndication-feed>` element, as shown in [Listing 12-4](#):

- `<search-type>pubmeta</search-type>`
- `<search-type>vermeta</search-type>`

These search types direct the Syndication Producer Servlet to query data from either a published (`pubmeta`) or versioned (`vermeta`) repository. The default search type is `pubmeta`.

Depending on the search type, either the `node` or `versioned` search property is set in the syndicated feed JSP ([Listing 12-2](#)). Specifically, the `node` or `version` is set to not null.

- `pubmeta` search type—`if (request.getAttribute("node") != null){`
- `vermeta` search type—`else if (request.getAttribute("version") != null){`

The `vermeta` search type is most often used in a workflow. You can only use this search type in a managed repository, that is, a repository where a library services are enabled. For more information, see “[Working with BEA Repository Content When Using Library Services](#)” on [page 10-3](#).

Syndication Configuration Options

The syndication configuration file for the preconfigured syndicated feeds offer a number of options. For example, you can change the order of results from descending to ascending. The available parameters are based on the XML schema shown in [Listing 12-6](#). The following options are available:

Table 12-2 Available Parameters for Passing into the Syndication Producer Servlet

Parameter	Description
Content Queries —use these parameters to find the information to be displayed in the syndicated feed.	
feedName	The name of an RSS feed stored within <code>wlp-syndication-config.xml</code> .
syndicationStyleName	The name of the format to use, such as RSS, Atom, and so on. Stored within <code>wlp-syndication-config.xml</code> .
searchQuery	The content search expression.
searchType	The type of search to execute: <ul style="list-style-type: none"> • <code>pubmeta</code>—published metadata search (default) • <code>vermeta</code>—versioned metadata search
searchSortOrder	Sort order of search results.
searchMaxResults	The maximum number of results to return from a search.
nameSpace	The content template name space, which is used to look up content templates for both <code><content-name-space></code> and <code><template-name-space></code> within the <code>wlp-template-config.xml</code> . If not provided, defaults to <code>bea-rss</code> .
RSS Channel Related Parameters —use these parameters to find the content template for creating the RSS channel header.	
channelGroupName	Content template group name.
channelTemplate	Content template name.
channelView	View name of the content template.
RSS Item Related Parameters —use these parameters to find the content template for creating each RSS item.	
itemResourceName	Content template resource name.

Table 12-2 Available Parameters for Passing into the Syndication Producer Servlet (Continued)

itemRepositoryName	Content template repository name.
itemView	Content template view name.

Selecting a Display Template

The `wlp-syndication-config.xml` file contains entries that map to the display template JSPs in the `wlp-template-config.xml` file. For example, in the `wlp-syndication-config.xml` file ([Listing 12-4](#)), the `syndication-style` elements point to corresponding elements (and JSPs) in the `wlp-template-config.xml` file ([Listing 12-5](#)).

From the `wlp-syndication-config.xml` file:

```
<syndication-style>
  <name>rss</name>
  <name-space>bea-rss</name-space>
  <channel-group-name>ContentRepoRSS</channel-group-name>
  <channel-template>RSSChannel</channel-template>
  <channel-header-view>channelHeader</channel-header-view>
  <channel-footer-view>channelFooter</channel-footer-view>
</syndication-style>
```

From the `wlp-template-config.xml` file:

```
<view>
  <name>channelHeader</name>
  <uri>/rss/rss_header.jsp</uri>
</view>
```

This design allows you to point to your own JSPs by making changes in the `wlp-template-config.xml` file. For example, to point to your header JSP you could change the URI from `/rss/rss_header.jsp` to `/atom/atom_header.jsp`.

To modify the preconfigured display template configuration file in Workspace Studio, from the Merged Projects View, copy the `WEB-INF\wlp-template-config.xml` file to your web project, and then edit the file. For information on how to use Workspace Studio for this purpose, see [Working with the Merged Projects View](#).

Listing 12-5 Example of a Display Template Configuration File (wlp-template-config.xml)

```

<?xml version="1.0" ?>
<wlp-template-config
xmlns="http://www.bea.com/ns/p13n/90/wlp-template-config">

<content-repository>
  <name>*/</name>
  <default-template-uri>/rss/rss_item.jsp</default-template-uri>
  <content-name-space>
    <name>bea-rss</name>
    <content-resource>
      <name>*/</name>
      <default-template-uri>/rss/rss_item.jsp</default-template-uri>
      <view>
        <name>detailedView</name>
        <uri>/rss/rss_detailed_view.jsp</uri>
      </view>
    </content-resource>
  </content-name-space>
</content-repository>
  <template-group>
    <name>ContentRepoRSS</name>
    <template-name-space>
      <name>bea-rss</name>
      <template>
        <name>RSSChannel</name>
        <default-template-uri>/rss/rss_header.jsp</default-template-uri>
        <view>
          <name>channelHeader</name>
          <uri>/rss/rss_header.jsp</uri>
        </view>
        <view>
          <name>channelFooter</name>
          <uri>/rss/rss_footer.jsp</uri>
        </view>
      </template>
    </template-name-space>
  </template-group>
</wlp-template-config>

```

Creating Custom Syndicated Feeds

It is likely that you want syndicated feeds customized to your organization. For example, you'll need to develop JSPs that display the portions of data in your content repositories that you want

to show your users, such as the name of the article and author's name. Additionally, you probably want to present the information in a format consistent with the look-and-feel of your portal. You can either use one of the preconfigured syndicated feed JSPs as a starting point or build one from scratch.

Tip: Be sure to read [“Using and Modifying the Preconfigured Syndicated Feeds”](#) on page 12-1 to help you understand how syndicated feeds work.

You can create syndicated feeds that are compatible with different readers, such as Atom. To speed your development process, you can use the same content query to retrieve information from your databases and present that content with different XML formats for different readers.

Because content repositories are connected to WebLogic Portal using the Virtual Content Repository, your JSP display templates can show data in any content repository as long as the repository is connected to the Virtual Content Repository. For more information, see [Chapter 1, “Introduction.”](#)

Creating the Syndicated Feed JSPs

Syndicated feed JSPs contain the logic to render the syndicated feed.

Generally, you need to create four JSPs:

- Header – Shows the title of the feed
- Item – Lists each article
- Footer – Closes the list
- Detail – Displays the content of an individual item in the list

For information on using Workspace Studio to create JSPs, see [Tutorial: Working with JSPs](#).

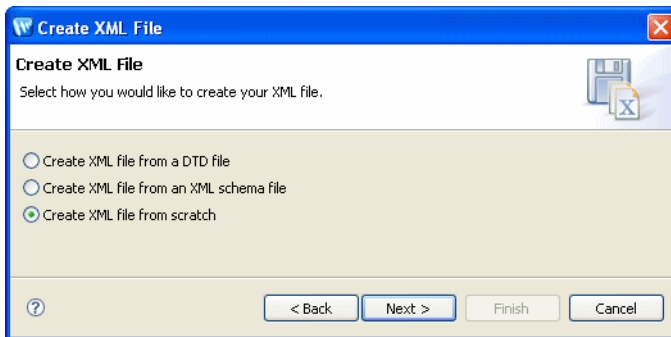
After you develop your display templates (JSPs), you map them to a namespace using a display template configuration file (`wlp-template-config.xml`). Display templates are then referenced from the syndication configuration file (`wlp-syndication-config.xml`).

Map the Syndicated Feed JSPs to Display Templates

To map your display templates to a namespace, you create a `wlp-template-config.xml` file. Display templates are then referenced from a syndication configuration file (`wlp-syndication-config.xml`). For detailed information on how this works, see [“Selecting a Display Template” on page 12-12](#).

For information about creating a `wlp-template-config.xml` file, see [“Creating a wlp-template-config.xml File” on page 11-3](#). When creating a template configuration file in this way, in the Create XML File page, select **Create XML file from scratch**.

Figure 12-3 Create XML File



For more information about display templates, see [Chapter 11, “Using Display Templates.”](#)

Create a Syndication Configuration File

The syndication configuration file sets the parameters passed to the Syndication Producer Servlet and maps the Syndication Feed JSPs to a namespace, which allows you to select which display templates are used. For more information, see [“Selecting a Display Template” on page 12-12](#).

[Listing 12-6](#) shows the XML schema for `wlp-syndication-config.xml`. It contains the valid attributes and tags you can use when configuring the `wlp-syndication-config.xml` file.

Listing 12-6 Schema for the Syndication Configuration File

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="http://www.bea.com/ns/content/102/wlp-syndication-config"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:complexType name="syndication-styleType">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="name-space" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="channel-group-name" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="channel-template" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="channel-header-view" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="channel-footer-view" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="item-resource-name" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="item-repository-name" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="item-view" type="xs:string" minOccurs="0"
      nillable="true"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="syndication-configType">
  <xs:sequence>
    <xs:element name="syndication-feed" maxOccurs="unbounded"
      type="wlp:syndication-storeType" minOccurs="0" nillable="true"
      xmlns:wlp="http://www.bea.com/ns/content/102
/wlp-syndication-config"/>
    <xs:element name="syndication-style" maxOccurs="unbounded"
      type="wlp:syndication-styleType" minOccurs="0" nillable="true"
      xmlns:wlp="http://www.bea.com/ns/content/102/
wlp-syndication-config"/>
    <xs:element name="syndication-item-style" maxOccurs="unbounded"
      type="wlp:syndication-item-styleType" minOccurs="0" nillable="true"
      xmlns:wlp="http://www.bea.com/ns/content/102/
wlp-syndication-config"/>
  </xs:sequence>
</xs:complexType>

```

```

<xs:complexType name="syndication-storeType">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="search-query" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="search-type" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="search-max-results" type="xs:int" minOccurs="0"
      nillable="true"/>
    <xs:element name="search-sort-order" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="name-space" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="channel-group-name" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="channel-template" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="channel-footer-view" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="channel-header-view" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="item-resource-name" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="item-repository-name" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="item-view" type="xs:string" minOccurs="0"
      nillable="true"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="syndication-item-styleType">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="name-space" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="item-resource-name" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="item-repository-name" type="xs:string" minOccurs="0"
      nillable="true"/>
    <xs:element name="item-view" type="xs:string" minOccurs="0"
      nillable="true"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="wlp-syndication-config" type="wlp:syndication-configType"
  xmlns:wlp="http://www.bea.com/ns/content/102/wlp-syndication-config"/>
</xs:schema>

```

Securing Syndicated Feeds

Usually, you control visitor access to portal resources by configuring visitor entitlements in the WebLogic Portal Administration Console. However, a malicious user who knows the correct URL can access the syndicated feed directly. If you want to secure your syndicated feeds, you must use Java EE security. Specifically, you use `web.xml` deployment descriptors to secure access to the Syndication Producer Servlet.

In following example, the `web.xml` deployment descriptors restrict any HTTP GET or POST requests from a URL in the form `/SecureFeedProducer/*`. These descriptors allow only users whose role is Administrators, Portal System Administrator, or AppTesters.

Listing 12-7 Security Deployment Descriptors

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Secure Syndication</web-resource-name>
    <description>The Secure Rss Feeds</description>
    <url-pattern>/SecureFeedProducer/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <description>Administrators</description>
    <role-name>Administrators</role-name>
    <role-name>PortalSystemAdministrator</role-name>
    <role-name>AppTesters</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

For more information about security deployment descriptors, see [web.xml Deployment Descriptor Elements](#) in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.

Part III Staging

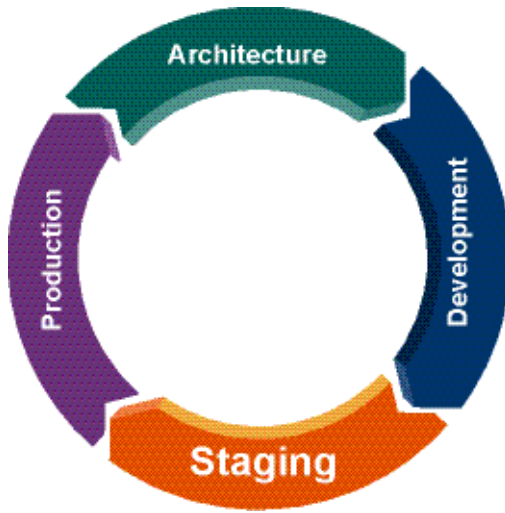
BEA content management repositories that are connected to the Virtual Content Repository are automatically staged with your portal application because they are stored in the portal database.

Part III contains information on managing workflows and includes the following chapter:

- [Chapter 13, “Managing Content Workflows in Your BEA Repository”](#)

For more information about staging and propagating content management repositories, see [WebLogic Portal Production Operations Guide](#).

For a description of the staging phase of the portal life cycle, see the [BEA WebLogic Portal Overview](#). The portal life cycle is shown in the following figure:



Managing Content Workflows in Your BEA Repository

You can manage content workflows in many of the same ways you manage content. You can add security features such as visitor entitlements and delegated administration to determine who can modify a workflow or associate a workflow with a content type or folder. For more information about visitor entitlements and delegated administration, see [Setting Delegated Administration on Content Management Resources](#) in the *WebLogic Portal Security Guide*.

To see which content workflows are in use, you can view which content types use which workflows, or view a list of all the content associated with a single workflow.

Note: You cannot delete a content workflow if it is currently assigned to a content type or any content.

Understanding Assigned Items

An item is assigned to you in two cases:

- When the workflow action assigns the item to a role to which you belong.

For example, suppose you have a draft content item in your Assigned Items folder. If you check out the item, it is moved to your Checked-Out Items folder. When you check in the item and you change its status to Ready for Approval, the item is removed from your Checked-Out Items folder and placed in the Assigned Items for an approver role. If the item is rejected, the item goes back to your Assigned Items folder. If the item is approved, it is published and will not return to your Assigned Items folder.

- An item may remain assigned to you if the workflow does not change the assignment.

For example, suppose you have a draft content item in your Assigned Items folder. If you check out the item, it is moved to your Checked-Out Items folder. When you check in the item and you change its status to Ready for Approval, the item is removed from your Checked-Out Items folder and placed back into your Assigned Items if you also belong to an approver role. If the item is rejected, the item goes back to your Assigned Items folder. If the item is approved, it is published and will not return to your Assigned Items folder.

Viewing Assigned Content Types

You can search for and view content types that are associated with workflows other than the default WebLogic Portal workflow.

To view content types that are associated with a content workflow other than the default workflow:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, click the workflow you want to view.
4. Click the **Associated Types** tab. Associated types appears in the **Browse Types Associated with this Workflow** section.

Viewing Assigned Content

You can view content that has been explicitly assigned a content workflow that overrides the content workflow associated with its type. For example, if some of the image content you have in your repository has been assigned a different workflow than the workflow associated with the “image” content type, you can use this feature to view that content.

To view content assigned to a particular workflow:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, click the workflow you want to view.

4. Click the **Associated Content** tab. Assigned content appears in the **Browse Content Associated with this Workflow** section.

Modifying a Content Workflow

You can modify a content workflow. When you modify a content workflow, you are modifying the workflow for all content and folders associated with that workflow.

To modify a content workflow:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, click the workflow you want to modify.
4. In the Details tab, optionally, click **Download File** if you want to modify the existing workflow document.
5. Click **Replace File** if you have already created a new workflow document and want to upload the new file.
6. In the Upload Workflow dialog, click **Browse** to select to the workflow document you want to upload, and then click **Open**.
7. In the Workflow dialog, click **Replace File**.

Deleting a Content Workflow

You can delete a content workflow if it is not associated with any content, folder, or content type.

To delete a content workflow:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. In the resources tree, right-click the workflow you want to delete.
4. Right-click the workflow and select **Delete**.

In the Delete dialog, click **Delete** to confirm the deletion.

Managing Content Workflows in Your BEA Repository

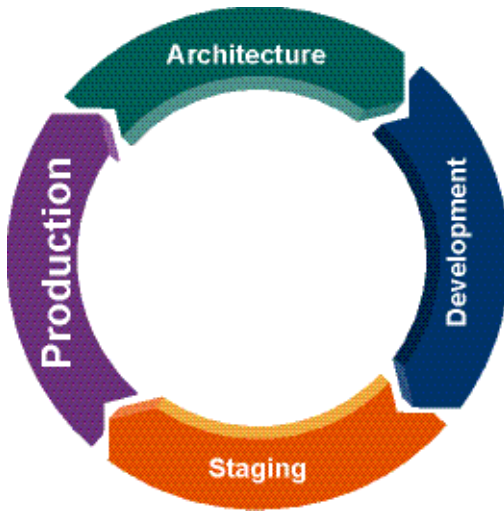
Part IV Production

This section contains guidelines and procedures to manage your content after you have deployed your portal and are running in production.

Part IV contains instructions for tasks you can accomplish in the production phase and includes the following chapters:

- [Chapter 13, “Managing Content Workflows in Your BEA Repository”](#)
- [Chapter 14, “Content Deployment Descriptor”](#)
- [Chapter 15, “Caching Content”](#)
- [Chapter 16, “Managing Content Security”](#)

For a description of the production phase of the portal life cycle, see the [BEA WebLogic Portal Overview](#). The portal life cycle is shown in the following figure:



Content Deployment Descriptor

The basic deployment configuration for an application is defined in a deployment plan. A deployment plan is a collection of deployment descriptors, which are XML documents. The deployment descriptors for content management are defined in the `content-config.xml` file.

Caution: WebLogic Server imposes no restrictions on the configuration properties that a user can modify using the WebLogic Portal Administration Console. Changes made in the administration console can override the settings made in the `content-config.xml` file. Therefore, if you use this method, be sure to follow the recommendations made in [Configuring Applications for Production Deployment](#) in *Deploying Applications to WebLogic Server*. It is recommended that you use the content management API's or administration tools to change your repository configuration.

This chapter contains the following topics:

- [Enabling Library Services in content-config.xml](#)
- [Setting User Credentials for an External Repository in content-config.xml](#)
- [Editing the content-config.xml File](#)

Enabling Library Services in content-config.xml

You can enable library services in `content-config.xml`. This is useful when you want to add a custom workflow that you want propagated in a production environment or when you want to share library services across a team of developers.

To enable library services using a deployment descriptor from WorkSpace Studio:

1. Follow the steps in [“Editing the content-config.xml File” on page 14-3](#) to access the `content-config.xml` file.
2. Using the WorkSpace Studio editor or other editor, add the following repository property to the `content-config.xml` file:

```
<repository-property>
  <name>MANAGEMENT_ENABLED</name>
  <value>>true</value>
</repository-property>
```

Setting User Credentials for an External Repository in content-config.xml

This section provides information about setting a cleartext password in `content-config.xml` when using a third-party repository.

To set a cleartext password in `content-config.xml`:

1. Follow the steps in [“Editing the content-config.xml File” on page 14-3](#) to access the `content-config.xml` file.
2. Using the WorkSpace Studio editor or other editor, add the following repository property to the `content-config.xml` file:

```
<content-store>
  <name>BEA_ExtendedRepository</name>
  <description>Portal Test Extended Repository
    Configuration</description>
  <class-name>com.bea.content.spi.internal.ExtendedRepositoryImpl
    </class-name>
  <password></password>
</content-store>
```

3. Start the server for the domain, if it is not already running.

4. Run the `webllogic.security.Encrypt` utility. For information on how to do this, see [Using the WebLogic Server Java Utilities](#) in *WebLogic Server Command Reference*.
5. Copy the generated domain-secret-encrypted-password into the password element. For example:

```
<password>{3DES}85Kmho5Uphc=</password>
```

Note: Password must begin with a {3DES} and end with =.

The password is specific to the given domain and must be run as a command line utility from that domain directory. This password will not work with another domain.

6. Restart the server.

Editing the content-config.xml File

To edit the `content-config.xml` file from WorkSpace Studio:

1. In the Project Explorer, navigate to the `<PortaLEARProject>/EarContent/META-INF` folder.
2. If `content-config.xml` is not in this folder, otherwise go to [step 3](#):
 - a. Select **Window > Show View > Other**.
 - b. Expand the WebLogic Portal node in the tree and click to select **Merged Projects View**.

Tip: Some items listed in the Merged Projects view are italicized. The italicized items represent entities that are stored in shared J2EE libraries. All entities that are stored on your file system are shown in regular type.

- c. Click **OK**.
 - d. In the Merged Projects View, right-click the `<PortaLEARProject>/EarContent/META-INF/content-config.xml` file and select **Copy To Project** from the context menu.
3. Double-click the **content-config.xml** file.
4. Using the WorkSpace Studio editor or other editor, edit the `content-config.xml` file.

Content Deployment Descriptor

Caching Content

After you have moved your portal application into production, you can:

- Use the WebLogic Portal Administration Console to edit and maintain content within the content repository
- Update personalization features such as content selectors and placeholders
- Change your repository caching

For information about adding and modifying content, see [Chapter 10, “Adding Content to a BEA Repository”](#).

For information about modifying content selectors or placeholders, see the *WebLogic Portal Interaction Management Guide*.

Caching content can improve the performance of your portal. You can create a cache of recently accessed content that can be quickly retrieved. You can increase the cache settings or decrease them as necessary. Cache settings are configured on a repository-basis.

To modify a cache settings for a repository:

1. From the main menu of the WebLogic Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. Click the repository you want to modify.
4. In the Advanced section, click **Advanced**.

Caching Content

5. In the Edit Advanced Properties for Repository dialog, modify binary and node caches as required.
 6. When finished making changes, click **Save**.
- A summary of the edited repository information is displayed in the Summary page.

Managing Content Security

WebLogic Portal provides two ways to secure your content: delegated administration and visitor entitlements. Delegated administration controls who can edit content within your repository, while visitor entitlements control who can view content when it is displayed in your portal.

This chapter includes the following sections:

- [Using Delegated Administration for Content](#)
- [Setting Visitor Entitlements for Content](#)

Using Delegated Administration for Content

Before starting to use the Virtual Content Repository, you can set up users, roles, and privileges to determine who can add, edit, or delete content. You use delegated administration roles to specify the users and groups that grant or deny access to resources and define which capabilities on those resources are available to administrators.

Before assigning content management resources to a delegated administration role, you must make sure the role exists. If no role exists, you must create it, then you can add users and content resources to it.

Setting Delegated Administration

You can set delegated administration rights at the repository level, folder level, or content item level.

If you set delegated administration rights at the repository or folder level, all children of that level inherit that delegated administration right. You cannot override a delegated administration right once it has been given. For example, if you give a delegated administration role rights to update content within a folder, you cannot remove that policy on an individual content item within the folder.

For detailed information about setting up delegated administration on content resources, see the [WebLogic Portal Security Guide](#).

Setting Visitor Entitlements for Content

Visitor entitlements allow you to define who can access the content in your portal application and what they can do with it.

For more information about visitor entitlements on content resources, see the [WebLogic Portal Security Guide](#).

Importing Third-Party Content

The BulkLoader is a command-line application that loads content and metadata from a file system into a BEA Virtual Content Repository.

Specifically, the BulkLoader scans a directory structure containing content and loads it into a specified content repository. In addition to loading content, BulkLoader reads prepared metadata files and associates the metadata with each loaded content item. Metadata files can be prepared for specific content items, or more broadly for directories and subdirectories of items.

Note: The BulkLoader only supports uploading of simple metadata and binary files. It does not support all BEA repository features, such as nested types and link properties.

If you use the BulkLoader to load content into a database repository, both the metadata and binary files are transferred to the repository. If you load into a file system repository, then only the metadata is transferred to the database—the actual content files remain in place on the file system.

Note: You cannot use the Bulkloader to update existing content (or its metadata) within a file system repository. It can only be used to add new content to the repository. To modify existing content, as well as adding new content, you use the WebLogic Portal Administration Console.

This chapter explains how to use BulkLoader and includes these topics:

- [Preparing to Use BulkLoader](#)
- [Configuring and Running BulkLoader](#)
- [BulkLoader Parameter Reference](#)

Preparing to Use BulkLoader

Before running BulkLoader, you need to:

- Create a repository
- Create appropriate content types
- Populate a directory structure with content
- Prepare metadata files

Creating a Repository

BulkLoader loads content and metadata into a pre-established content repository. For information on creating a repository, see [“Configuring BEA Repositories” on page 3-1](#).

Creating Appropriate Types

Each piece of content stored in a repository is associated with a *type*. A type is a definition that includes specific metadata fields that can be used to identify and describe content items associated with that type. The BEA Repository contains several predefined default types. For example, the predefined *image* type contains three metadata fields:

- Description
- File
- Image Name

You can create your own types or use the provided types. For information on creating types, see the [“Using Content Types in Your BEA Repository” on page 6-1](#).

Note: A type associated with a content item must be created with *binary* as its primary property.

Preparing a Content Directory

BulkLoader loads all the content from a specified directory (and, by default, subdirectories) into the content repository. Directories are automatically recreated as hierarchy nodes (folders) in the content repository. The directory structure you load into the repository should only contain the content you wish to add to the repository. BulkLoader loads all files within this directory structure.

Tip: You can configure BulkLoader, using command-line flags, to ignore or include particular files or folders based on filename pattern matching.

Preparing Metadata Files

Each piece of content in the repository is mapped to a specific type. A type includes default and user-defined properties. These properties, also known as metadata, allow content items in the repository to be identified and searched.

BulkLoader allows you to automatically associate individual files and/or directories of files with specific types. This section describes these associations and how to add metadata when library services are enabled for your repository, plus how to properly name and store metadata files.

Defining Metadata for a Directory of Files

If you know that an entire directory (and, by default, its subdirectories) contains files of the same type, you can specify that type to be associated with all of those files when BulkLoader stores them in the repository. To do this, place a file called `dir.md.properties` in the root directory containing the related content. This file must contain a single line:

```
nodeType=type
```

where *type* is the name of the type to associate with the content. For example:

```
nodeType=image
```

By default, all content in the directory and its subdirectories will be associated with the type. If a subdirectory contains another `dir.md.properties` file, then the type defined in that file overrides the original one for that directory and any of its subdirectories. Furthermore, if a `filename.md.properties` file is encountered, it also overrides the `dir.md.properties` file for that specific file. The `filename.md.properties` file is described next.

Defining Metadata for Specific Files

You can also define metadata for specific files loaded by BulkLoader. To do this, create a file called `filename.md.properties` for each piece of content, where `filename` is the name of the file with which the metadata is associated. This file must contain all of the name-value pairs associated with a type. For example, the following entries are associated with the Ad type:

```
nodeType=Ad
height=65
width=115
adTargetUrl=
adTargetContent=
adWinClose=
adWinTarget=
adWinTitle=
adClickTarget=
adUseXhtml=
adAltText=BEA Logo
adMapName=
adMap=
adBorder=
audience=internal
```

You can then add values for some or all of these properties and save the file. Place the saved file in the same directory as the content item with which it is associated. When BulkLoader runs, the metadata is stored and permanently associated with the specified content item.

Metadata Guidelines

- A content type may have only one binary property, and it must be marked as the primary property.
- If a type has required fields, those fields must be given values in the `filename.md.properties` file.
- If you are bulkloading for a file system repository, only one binary property is allowed, and it must be the primary property.
- When uploading DATE/TIME properties, you need to use the `java.text.DateFormat.SHORT` format which is `MM/DD/YY HH:MM AM/PM`. The order of the day/month in the date is dependent on the locale of the JVM.

Creating Metadata for a Library Services Enabled Repository

If you are storing content in a library services enabled repository, you must include the `lifecyclestatus` key in the `filename.md.properties` file for each content item. The `lifecyclestatus` key takes the following integer values that indicate the status of the content item, as shown in [Table A-1](#).

Table A-1 Workflow Status

Workflow Status	Integer Used
Draft	1
Ready	2
Published	4
Retired	5

For example, the following `md.properties` entries are associated with the Ad type, and include the `lifecyclestatus` entry, where the status value is set to 2, or “ready”.

```
nodeType=Ad
height=65
width=115
adTargetUrl=
adTargetContent=
adWinClose=
adWinTarget=
adWinTitle=
adClickTarget=
lifecyclestatus=2
adUseXhtml=
adAltText=BEA Logo
adMapName=
adMap=
adBorder=
audience=internal
```

You can then add values for some or all of the other properties and save the file. Place the saved file in the same directory as the content item with which it is associated.

Note: If you are bulkloading content into a library services-enabled repository, you can only add content. You cannot update existing content with the BulkLoader when using library services.

Naming and Storing Metadata Files

When BulkLoader encounters a directory to process, it attempts to load metadata property files. First, BulkLoader looks for a file called `dir.md.properties` in the directory. If there are no overriding metadata files, these properties are applied to all content items in the directory and, unless overridden, its subdirectories. Metadata files can be associated with specific content files, and these metadata files override the directory-level file. Metadata files associated with specific content files must be named according to the following convention:

`filename.md.properties`

where *filename* is the name of the associated content item file. For example:

`logo.gif.md.properties`

In this case, the metadata file is associated with an image file called `logo.gif`.

Note: You can change the default extension from `md.properties` to anything you like, using BulkLoader's `-mdext` parameter.

Tip: By default, BulkLoader recurses into subdirectories and the properties in an `dir.md.properties` file are inherited by content in subdirectories. To override this behavior, specify the `+recurse` flag, which turns off recursion and specify the `+inheritProps` flag, which turns off metadata property inheritance in subdirectories.

Metadata Summary

In summary, BulkLoader gathers content metadata from the following sources, in the order shown:

- The `dir.md.properties` file in a parent folder.
- The `dir.md.properties` file in a subfolder.
- A `filename.md.properties` file (applied to a specific file).
- The `<meta>` tags in an HTML file. For more information, see the description of the `htmlPat` flag in the section [BulkLoader Parameter Reference](#).

- The list of LoadFilters. For more information, see the description of the `filter` flag in the section [BulkLoader Parameter Reference](#).

Configuring and Running BulkLoader

Typically, you run BulkLoader from a script. You need to edit this script to run in your environment, and to customize parameters that are passed to the BulkLoader program itself.

Note: If BulkLoader fails with an out of memory error, increase your Java heap size. You may do this in the BulkLoader script by passing `-Xms<xxx>m` as a parameter to the BulkLoader command, where `<xxx>` is the number of megabytes. For example, `-Xms1000m`.

The following script is provided with Weblogic Portal:

Windows: `<BEA_HOME>\wlserver_10.0\cm\bin\load_cm_data.cmd`

Unix: `<BEA_HOME>/wlserver_10.0/cm/bin/load_cm_data.sh`

Note: WebLogic server must be running when you use BulkLoader.

Editing the BulkLoader Script

You need to modify the default script to match your needs. The following script is provided with Weblogic Portal:

Windows: `<BEA_HOME>\wlserver_10.0\cm\bin\load_cm_data.cmd`

Unix: `<BEA_HOME>/wlserver_10.0/cm/bin/load_cm_data.sh`

1. Open the script file for editing.
2. Set the `WL_HOME` variable to point to your WebLogic Server installation. For example:


```
WL_HOME=C:\bea\wlserver_10.0
```
3. Set the `CM_DATA` variable to point to the parent directory of the directory containing the content you wish to load into the content repository. For example, if the content you want to store is located in `D:\myContent\images`, set `CM_DATA` to:


```
CM_DATA=D:\myContent
```
4. Configure the BulkLoader command parameters in the command script. [Listing A-1](#) shows a sample configuration.

Listing A-1 Example Bulkloader Script

```
%JAVA_HOME%\bin\java -classpath %CLASSPATH%  
com.bea.content.loader.bulk.BulkLoader -verbose -repository "MyRepository"  
-application portalApp -d %CM_DATA% file1 file2 filen
```

The parameters shown in bold type are described in [Table A-2](#).

Table A-2 Bulkloader Parameters

Parameter	Description
-verbose	Prints messages while BulkLoader is running.
-repository	Specifies the name of the repository into which you are loading content.
-application	Specifies the name of the Portal application with which the repository is running.
-d	Specifies the base directory that contains the directories and files you wish to load into the content repository. If you do not specify this option, the current directory (.) is used. When bulkloading content to a file system repository, you must specify this option and the directory must match the <code>cm_fileSystem_path</code> property that was defined when the repository was created. The path specified can be relative or absolute. For more information about file system repositories, see “Working with a BEA File System Repository” on page 3-9.
file1...filen	The name(s) of the files and/or folders to load into the content management system. These files and folders are assumed to be located relative to the base directory (-d).

For a description of all BulkLoader parameters, see [BulkLoader Parameter Reference](#).

Tip: You can run the BulkLoader script from the command line or by double-clicking the file icon.

BulkLoader Command Examples

Note: The BulkLoader command does not support wildcards or regular expressions in its parameter list.

The following command recursively loads all files in the directories Images, Audio, and Doc in D:\media. Note that Images, Audio, and Doc must each contain a `dir.md.properties` file, or there must be a `filename.md.properties` file defined for each content item in those directories.

Listing A-2 Example of a BulkLoader Command Script

```
%JAVA_HOME%\bin\java -classpath %CLASSPATH%
com.bea.content.loader.bulk.BulkLoader -verbose -repository "MyRepository"
-application portalApp -d D:\media Images Audio Doc
```

The command shown in [Listing A-3](#) loads all files in D:\media\images. The command does not recurse into subdirectories. Metadata files with a `*.info.properties` naming convention are recognized.

Listing A-3 Example of a Bulkloader Command Script

```
%JAVA_HOME%\bin\java -classpath %CLASSPATH%
com.bea.content.loader.bulk.BulkLoader -verbose -repository "MyRepository"
-application portalApp -mdext info.properties +recurse -d D:\media images
```

BulkLoader Parameter Reference

Table A-3 Required Bulkloader Parameters

Required Parameters	Description
-repository <repository name>	The name of the repository to run the loader against.
-application <app name>	The name of the application to run the loader against.
-url <wls url>	The WebLogic Server instance host where the content manager is running.
-user <principal username>	The username for the principal permitted to access the Loader EJB resource.
-password <principal password>	The password for the principal permitted to access the LoaderEJB resource.
-d <dir>	Specifies the base directory that contains the directories and files you wish to load into the content repository. If you do not specify this option, the current directory (.) is used. This directory must match the <code>cm_fileSystem_path</code> property that was defined when the repository was created. The path can be relative or absolute.

Table A-4 Optional Bulkloader Parameters

Optional Parameters	Description
-recurse	Recurse into directories. (Default)
+recurse	Do not recurse into directories.
-metaparse	Parse HTML files for META tags. (Default)
+metaparse	Do not parse HTML files for META tags.
-hidden	Ignore hidden files and directories. (Default)
+hidden	Include hidden files and directories.
-inheritProps	Inherit metadata properties when recursing. (Default)
+inheritProps	Do not inherit metadata properties when recursing.

Table A-4 Optional Bulkloader Parameters (Continued)

Optional Parameters	Description
-ignoreErrors	Ignore errors while loading content; errors are still reported.
+ignoreErrors	If an error is encountered, stop preprocessing. (Default)
-htmlPat <pattern>	Specifies file extensions that represent HTML files. If the -metaparse flag is set, the values of <meta> and <title> tags are read from these files and stored as content metadata. You can specify this flag multiple times to define multiple file extensions. By default, *.htm and *.html are used.
-encoding <encoding>	Specifies the file encoding to use. See your JDK documentation for the valid encoding names. (Default: the system's default file encoding)
-match <pattern>	Specifies a file/directory name pattern for BulkLoader to load. All files matching this pattern are loaded. You can specify this flag multiple times to define more patterns. If this flag is omitted, all files and directories are loaded.
-ignore <pattern>	Specifies a file/directory name pattern for BulkLoader to ignore. All files matching this pattern are ignored. You can specify this flag multiple times to define more patterns.
-mdext <ext>	Specifies the file name extension for metadata property files. The value should start with a ".". This defaults to .md.properties.
-filter <filter class>	<p>Although not commonly used, you can write a custom filter to set metadata values based on specific characteristics of a type of content. For instance, a filter might compute the width and height of an image file and set the values in metadata.</p> <p>This flag sets the class name of a LoaderFilter to run files through. To add to the list of LoaderFilters, you can specify this flag multiple times. The LoaderFilter may assign additional metadata to the file. When BulkLoader starts up, it looks for a content\com\bea\content\loader\bulk file in the classpath. From there, it looks for a loader.defFilters property, which is the colon-separated list of LoaderFilter class names that BulkLoader should always load. Unless this file is modified, BulkLoader loads an ImageLoaderFilter, which pulls the width and height from *.gif, *.jpg, *.png, and *.xbm image files.</p>
-filters	Clears the current list of LoaderFilters, including the default filters.

Table A-4 Optional Bulkloader Parameters (Continued)

Optional Parameters	Description
--	Everything after this is considered to be a file or directory to be uploaded.
-Xms<xxx>m	Increases the Java heap size, where <xxx> is the number of megabytes. For example -Xms1000m. Try using this flag if BulkLoader fails with an out-of-memory error.
-h	Display command line usage.

Example BulkLoader Script

[Listing A-4](#) configures the appropriate paths and runs the BulkLoader program. You can modify this script to suit your specific environment and requirements.

Listing A-4 BulkLoader Script (Windows)

```

@ECHO OFF
REM
#####
REM #      (c) BEA SYSTEMS INC. All rights reserved
REM #
REM # NOTE: WL_HOME must be set before running this script
REM
#####

SETLOCAL

echo
echo ***** NOTE: The environment variable WL_HOME must be set before running
this script
echo

SET CONTENT_HOME=%WL_HOME%\cm
SET P13N_HOME=%WL_HOME%\platform\lib\p13n
SET
P13N_JARS=%P13N_HOME%\p13n_system.jar;%WL_HOME%\common\p13n\lib\p13n_app.jar

SET CONTENT_JARS=%CONTENT_HOME%\lib\content-client.jar;%CONTENT_HOME%\lib
\content.jar;%WL_HOME%\platform\lib\cm\content_system.jar

CALL %WL_HOME%\common\bin\commEnv.cmd
    
```

```
@rem
*****
@rem Set any additional CLASSPATH information below
@rem
*****

set CLASSPATH=%WEBLOGIC_CLASSPATH%;%P13N_JARS%;%CONTENT_JARS%

echo Running script with CLASSPATH: %CLASSPATH%
echo

REM Set some defaults
if "%CM_DATA%"==" " set CM_DATA=..\db\data\sample\cm_data

@rem These are the deployed app settings
%JAVA_HOME%\bin\java -classpath %CLASSPATH%
com.bea.content.loader.bulk.BulkLoader -verbose
-repository "Shared Content Repository" -application "portalApp"
-d %CM_DATA% Ads%*

ENDLOCAL
```

Importing Third-Party Content