BEA WebLogic Portal®

Content Management Guide

Version 10.0
Revised: March 2007

Contents

3. Configuring BEA Repositories

4. Using Content Folders in Your BEA Repository

5. Using Content Workflows in Your BEA Repository

7. Using WebDAV with Your BEA Repository

6. Using Content Types in Your BEA Repository

A. Importing Third-Party Content

CHAPTER 1

Introduction

Most portals incorporate content into their applications. Content can be anything from advertisements (graphic files), documents (word processor files), or animation files. Content that you use in your portal is typically stored in a content repository that is part of a content management system that is connected to your 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

BEA WebLogic Portal®'s content management system 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 WebLogic Portal's repositories to store and access content. If enabled, you can content contributors can also access content through content applications such as Microsoft

Word. This is accomplished through WebDAV. For more information, see Chapter 7, "Using WebDAV with Your BEA Repository."

Content repositories are connected to your portal by using 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.

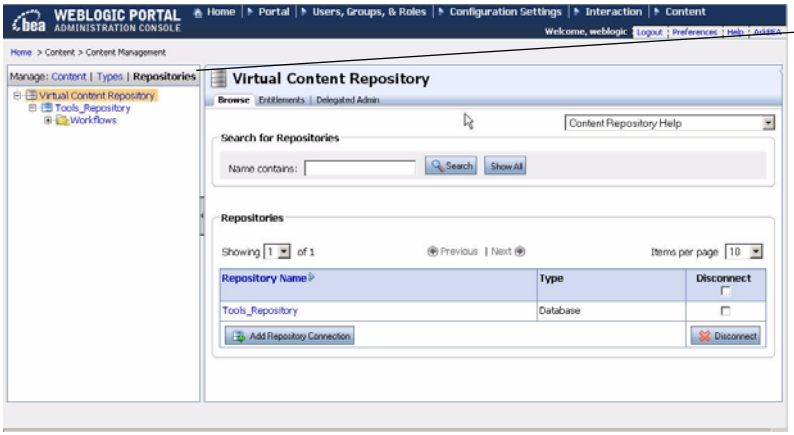
Storing Content

Content repositories are connected to WebLogic Portal via WebLogic Portal's 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 such 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 WebLogic Portal's 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

The Virtual Content Repository is typically accessed through the Portal Administration Console, see 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.**

This section includes the following topics:

Viewing the Virtual Content Repository

The Virtual Content Repository allows you to view your content repositories in three ways: 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**)	• Add content to the repository • Search for content • Delete content • Modify content If using BEA's library services, you can also: • View version history • Check out content • Check in content • For more information about library services, see "Overview of BEA's Library Services" on page 9-4.

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

Virtual Content Repository View	Associated Tasks
Types view (Click **Manage | Types**)	• Add content types • Modify content types (adding or removing property definitions) • Delete content types
Repository view (Click **Manage | Repositories**)	• Connect repositories to the Virtual Content Repository • Edit repository properties such as search settings and caches If using BEA's library services, you can also: • 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. You can use multiple BEA repositories, as well as configure them to use filesystem 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 further 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, and 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.

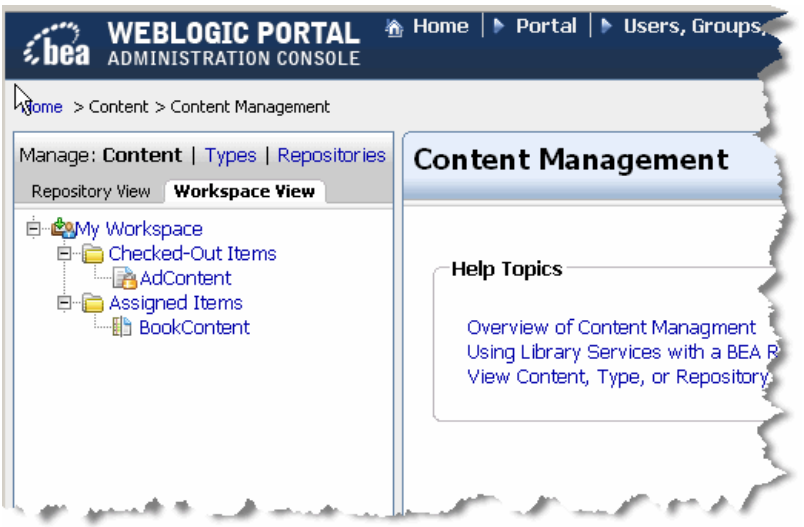
You can also enable BEA repositories to take advantage of BEA's library services. Library services allow users to track versions of content and use content workflows to enforce processes you want content contributors to use, including getting approval and retiring outdated content.

For more information about library services, see “Adding Content to a BEA Repository” on page 9-1.

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.
- Take advantage of user-specific workspaces that display which items need a user's approval at any time or show the status of the content the user created as shown in Figure 1-2.
- Use content version control which allows you to keep multiple versions of content within your database.

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



Adding Content

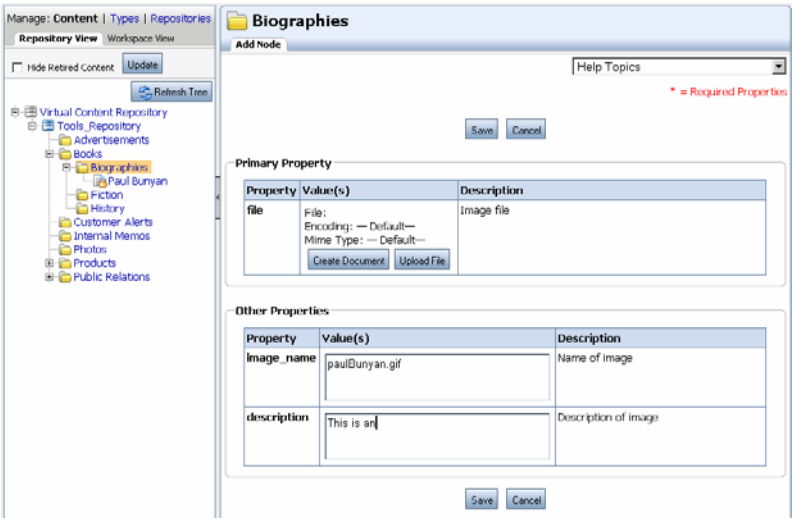
Typically, content contributors can add content to a BEA repository through the Portal Administration Console which provides access to your content repositories. Content users can also add content via Windows Explorer, if you have configured your repository to use WebDAV. For more information, see “Using 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. This metadata is used by portal developers 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 9-1.

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 that use rules to personalize content delivery for portal users. For more information about delivering content in your portal, see the Chapter 10, "Delivering Content Within Your Portal."

WebLogic Portal's content management system 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 WebLogic Portal's wide variety of personalization tools. Content selectors, placeholders, and campaigns deliver dynamic, personalized content to user based upon personalization rules or 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.

You can use Delegated Administration to determine which users can add or modify content within the Portal Administration Console. For example, you can allow only certain users to approve content for publishing, or disallow users from deleting content from your repository. Visitor Entitlements are used to allow or disallow portal visitors to view portal content. For more information about security, see the *WebLogic Portal Security Guide*.

Content Management in the Portal Life Cycle

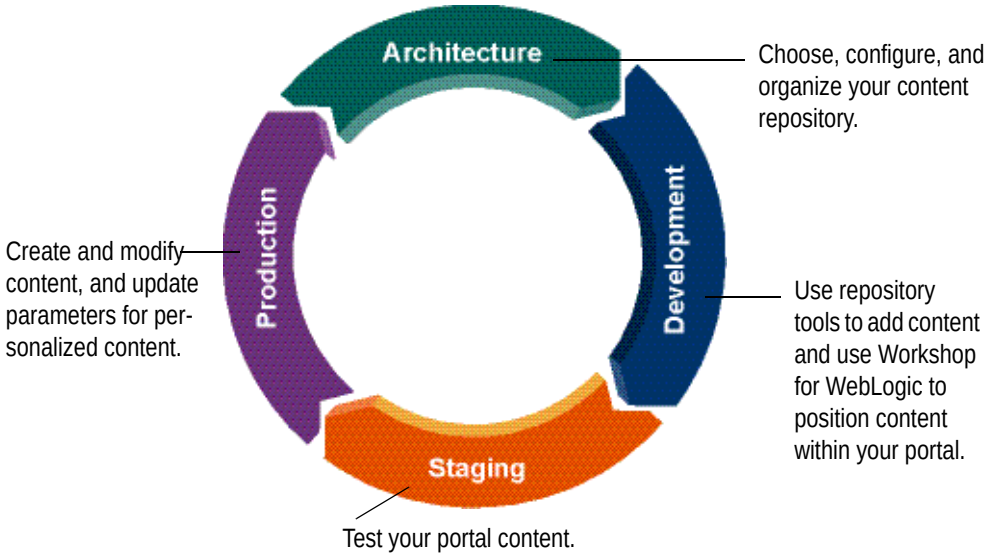
The tasks in this guide are organized according to the portal life cycle. For more information about the portal life cycle, see the *WebLogic Portal Overview Guide*. The portal life cycle contains four phases: architecture, development, staging, and production.

This section includes the following topics:

- 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 what type of content repository you will use and set it up to match your business needs. This includes creating content types to store content, creating content workflows to enforce process, and creating content folders to organize your repository. You also plan out propagation strategies and determine which content tools you will use.

The chapters describing tasks within the architecture phase are:

- 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”

Development

During the development phase, content contributors add content and developers determine how to present that content within your portal. Content contributors use the Portal Administration Console to add and manage content. Using Workshop for WebLogic, portal developers use content selectors, placeholders, JSP tags, HTML, and the content API to retrieve and display content. For more information about delivering content within your portal, see the *WebLogic Portal Interaction Guide.*

The chapters describing tasks within the development phase:

- Chapter 9, "Adding Content to a BEA Repository"

Staging

The staging phase is when test your portal and verify that developed content is appearing correctly. You might move iteratively between developing and then 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.

- Staging

Production

After you test your portal application in the staging phase, use the production phase to manage your production environment. During the production phase, you use the Portal Administration Console to adjust settings, add content, and modify content selectors or create ad campaigns. See the *WebLogic Portal Interaction Management Guide* for more information about content selectors or campaigns.

You can also set up security for your content using Delegated Administration and Visitor Entitlements, see the *WebLogic Portal Security Guide* for more information.

- Chapter 12, "Caching Content"

Part I Architecture

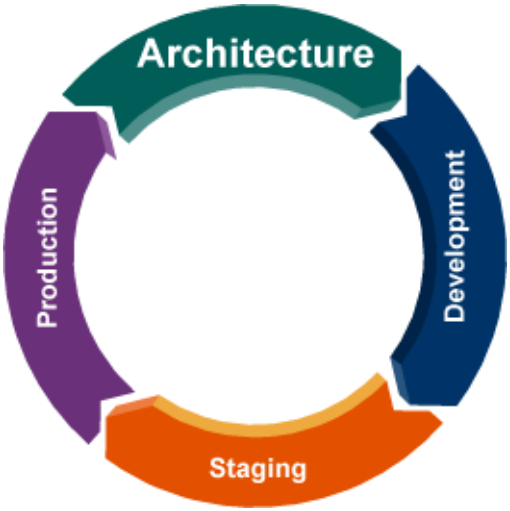
This part contains information to help you complete the architecture phase of using WebLogic Portal's content management features.

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 graphic:


Architecture
Development
Staging
Production

CHAPTER 2

Using Content Repositories

Before you can begin to develop content and incorporate 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 filesystem repository.

If you already have existing content in another repository, you can connect it to the Virtual Content Repository via an SPI (Service Provider Interface) implementation.

This chapter includes the following sections:

Connecting Repositories to the Virtual Content Repository

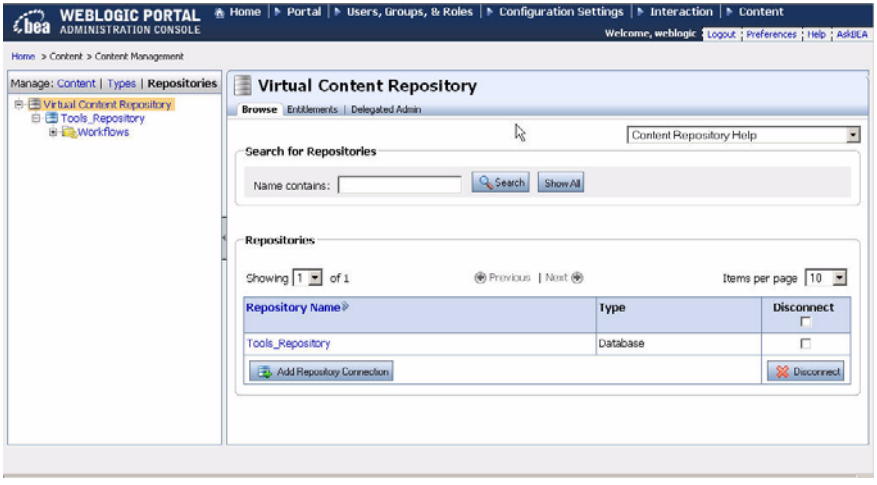
Regardless of what type of repository you use to store your content, all repositories must be connected to WebLogic Portal's Virtual Content Repository. The Virtual Content Repository allows content contributors to access repositories using the content tools in the Portal Administration Console. It also connects your repositories to your portal application so 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 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 WebLogic Portal's library services which provides content workflows and versioning. For more information about library services, see "Adding Content to a BEA Repository" on page 9-1.

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 Filesystem Repository

BEA filesystem repositories allow you to use a filesystem in tandem with the BEA database to store your content. When you use a filesystem repository, content binary files (stored as binary properties) are stored in the filesystem 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 filesystem.

Note: When you use a filesystem repository, it is important for content contributors to work with content using the Portal Administration Console. If you manipulate content within the filesystem directly, data will be out of sync and behave unpredictably.

When you are using a filesystem repository with library services, non-published versions of content are stored in the BEA database. This provides an additional safeguard if the filesystem gets damaged or removed. For more information about library services, see Chapter 9, "Adding Content to a BEA Repository."

For more information about using a filesystem 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 WebLogic Portal's 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 Portal Administration Console.

However, if your vendor has not implemented BEA's SPI, then you can use the instructions in Chapter 8, "Connecting to a Third-Party Repository," to do so yourself.

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

If the SPI implementation is read only, then you will need to continue 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 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. You set up a hierarchy of content folders and also create content types that define the content properties that content contributors provide values for when creating content, such as the name of the content, the author, and so on.

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 out your content types and content folders, you can 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 or whether portal users can click it. Another example may be an "start date" property for content so developers can schedule content for go-live at a future date, as well as content expiration.

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

You can set up the users, roles, and capabilities you want to use in order to dictate who can add, edit, or delete content within the repository.

Visitor Entitlements allow a way for portal administrators to limit what content portal users can see. For example, within a portal used for human resources, 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 that will create content or maintain your repository within the Portal Administration Console. For example, you can set up a role that is used only by users who are responsible for publishing content, where that role will only see the features related to creating and editing content, but none of the administrative features such as creating types or workflows. You can also create a separate role for creating content types or new folders.

Users and roles are used when you set up Delegated Administration as well as Visitor Entitlements. 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*.

CHAPTER 3

Configuring BEA Repositories

Each WebLogic Portal is pre-configured with a BEA default BEA repository for you to use. The default BEA repository can be library services-enabled which 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 you choose to use 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 configure a BEA filesystem repository. Both 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)	• Pre-configured out-of-the-box. • Library services built-in, such as 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 Filesystem Repository	• Stores published content in a filesystem. • Library services built-in, such 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 filesystem. **Note:** Does not support transactions as a database-based repository does. If your network connection goes down when adding or modifying content, changes could be lost.

Working with a Default BEA Repository

The default BEA repository comes pre-configured for your portal application. Before using your BEA repository, you need to ensure that library services are enabled and add any custom properties needed for your environment.

Some examples of repository properties include those needed to enable integration with features such as 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, see "Adding Content to a BEA Repository" on page 9-1 for more information about library services.

This section discusses the following topics:

- Enabling Library Services for a BEA Repository
- Modifying a BEA Repository

Enabling Library Services for a BEA Repository

WebLogic Portal's 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. Enabling library services gives you access to content workflows and versioning.

For more information about using library services, see Chapter 9, "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 Portal Administration Console, select **Content > Content Management**.
2. Select **Manage | Repositories** to view the Repositories tree.
3. Click the BEA repository for which you want to enable library services.
4. In the **Summary** tab, click **Library Services** to view the **Library Services** dialog.
5. Mark 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, for your repository.

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 the *WebLogic Portal Search Guide*.

To modify a repository connection:

1. Select **Content > Content Management**.
2. Select **Manage | Repositories**.
3. Click the repository you want to modify to view its Summary tab.
4. In the repository's Summary tab, use the following sections to modify your repository connection.

5. In the Repository Details section:
   – Click Repository Details to change the repository class name 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: • `com.bea.content.spi.internal.RepositoryImpl` If you are using a filesystem repository, the connection class is the following: • `com.bea.content.spi.internal.FileSystemRepositoryImpl` For third-party repositories, please see your vendor SPI documentation.

   – Click **Change Password** to change the password for this repository. Passwords are generally used for third-party repositories.

6. In the Library Services section, you can enable library services.
7. In the Properties section, you can add or modify repository properties, see "Adding Custom Properties" on page 3-4.
8. In the Advanced section, you can modify the repository cache settings, see "Editing Advanced Repository Properties" on page 3-6.

Adding Custom Properties

You can define properties for repositories within the Virtual Content Repository. For example, if you are using a BEA filesystem repository, additional properties should be added. Table 3-3 lists some examples of repository properties.

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

To add a property to a repository:

1. From the main menu of the Portal Administration Console, select **Content > Content Management**.
2. In the resource tree, click **Repositories** to view the Manage | Repositories tree.
3. In the Manage | Repositories resource tree, select the BEA Repository to which you want to add a property.

4. In the Properties section on 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.

Table 3-3 Examples of BEA Repository Properties

Property	Definition
`cm_fileSystem_webpath`	When using a filesystem repository that is available over the web, this property indicates the path of that filesystem.
`cm_fileSystem_webpath`	When using a filesystem repository that is available over the web, this property indicates the path of that filesystem if it is accessed via a web server.
`webdav_enabled`	Used when you want to enable 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 (Windows Explorer and other WebDAV supported applications). For more information about configuring WebDAV, see Chapter 7, "Using WebDAV with Your BEA Repository."
`cm_fireFederatedEvents`	Used to enable 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`	Used to enable events associated with content changes. This is used for full-text search. For more information about events, see the *WebLogic Portal Interaction Guide*.
`CM_DATA_SOURCE`	Used to associate a data source with a repository. This property is used when you have more than one BEA repository. For more information, see "Configuring Additional BEA Repositories" on page 3-10.

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 maximum number of content items that can be cached.
Node Cache	Determines the maximum number of content folders that can be cached.

To edit advanced repository properties:

1. Select **Content > Content Management** from the navigation menu at the top of the console.
2. Select **Manage | Repositories**.
3. In the resource tree, click the repository you want to modify to view its Summary tab.
4. In the Advanced section, click **Advanced** to view the Edit Advanced Properties for Repository dialog.
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, 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 need to delete the content store (database or filesystem). Deleting a datastore or database should be done by a database administrator.

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

To disconnect a repository:

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

Note: If a repository was added manually (not through the Portal Administration Console), you cannot disconnect it using the Portal Administration 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, see "BEA File System Repository Considerations" on page 3-7 for more information.

This section discusses the following topics:

- BEA File System Repository Considerations
- Configuring a File System Repository

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, there are

content requirements that help maintain the integrity of the repository and its associated filesystem.

- In a filesystem repository, you cannot associate folders with content types. For more information about content folders, see Chapter 4, "Using Content Folders in Your BEA Repository."
- The names of content item must match the name of the binary file associated with it. For example, a content item associated with a binary file called `ball.gif` must be named **ball.gif**.
- 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 filesystem.

Before You Begin

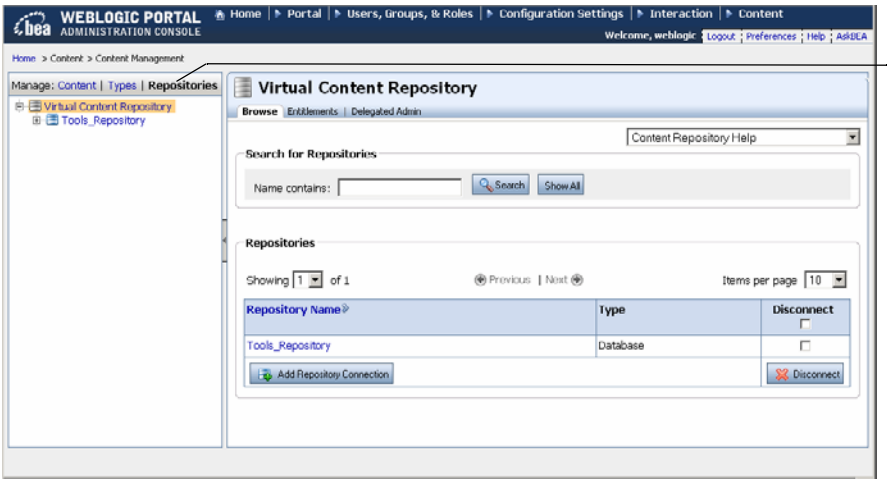
- The recommended way to implement a file system repository is to modify the properties of the default BEA repository.
- The file system or directory where your file system repository will reside must already exist before creating the repository connection.

Creating a Connection to the New Filesystem Repository

Using the Portal Administration Console, edit the connection information for the default BEA repository. If you want to create an additional repository, see "Configuring Additional BEA Repositories" on page 3-10 then return to this section.

1. Select **Content > Content Management** from the navigation menu.
2. View the **Manage | Repositories** tree by selecting **Repositories**. Figure 3-1 shows an example of selecting the Repositories area.

Figure 3-1 Manage Repositories Tree within the Virtual Content Repository



Select Repositories to work with a repository in the Virtual Content Repository.

3. In the **Manage | Repositories** tree, click the existing BEA repository to view the Repository Summary.
4. Click **Repository Details**.
5. In the **Repository Details** dialog box, 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 filesystem that contains your content. For example: `/home/myData`.
      - Be sure that each machine within your portal application cluster has network access to your filesystem.
   c. Click **Save.**
8. If your filesystem is exposed though a web server, you should also set a `cm_fileSystem_webpath` property. This enables the repository to access content through the web server that the filesystem uses.

To add this property, click **Add Property**.

a. In the **Key** field, type `cm_fileSystem_webpath`.

b. In the **Value** field, enter the URL of your filesystem.

Note: For example, your `cm_fileSystem_path` could be set to `/home/myData` but the same path could be referred externally as `http://mydomain.com/data/myData` which can be set using the `cm_fileSystem_webpath` property.

9. Click **Create**.

Your filesystem repository appears in the resource tree.

Configuring Additional BEA Repositories

You can create multiple content repositories within the Virtual Content Repository to meet your unique 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.

This section includes the following topics:

- Considerations for Additional BEA Repositories
- Creating Database Objects for the New Repository
- Connecting the New Repository to the Server
- Connecting the New BEA Repository to the Virtual Content Repository

Considerations for Additional BEA Repositories

- While multiple repositories can reside in the same database, each repository must store its data in a separate tablespace and have a unique name.
- If using library services, you must be working within an XA domain to effectively use multiple repositories.
- 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 is to ensure that each 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 steps:

- 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.

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 which 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/database user must be predefined according to the appropriate sample script, see the *WebLogic Portal Database Administration Guide* for more details.

   – For **Oracle**, BEA provides the following sample scripts: *`WebLogic_HOME`*`/portal/db/oracle/admin/create_tablespaces.sql` and `create_users.sql`.

   – For **SQL Server**, BEA provides the following script: *`WebLogic_HOME`*`/portal/db/sql_server/admin/create_database.sql`

   – For **Sybase**, BEA provides the following script: *`WebLogic_HOME`*`/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 via 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`*`/portal/db/db2/admin/create_tablespaces.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 which is called `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 which is called `database.properties`.

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

You can now connect your repository to WebLogic Server.

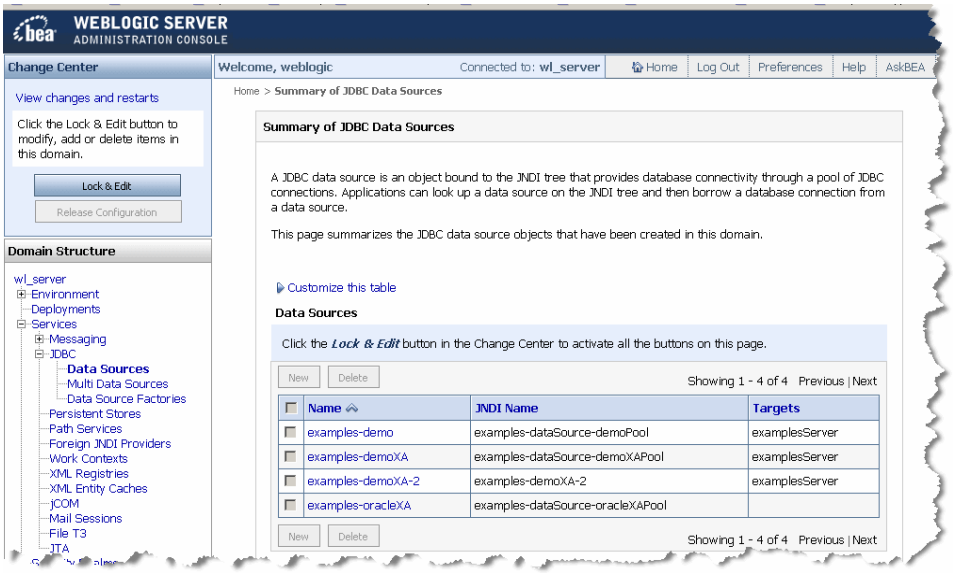
Connecting the New Repository to the Server

To connect the new repository to the server, you need to configure the WebLogic Server to point to the new repository. You do this by creating a new data source for the repository you want to use.

To connect a repository to the WebLogic Server:

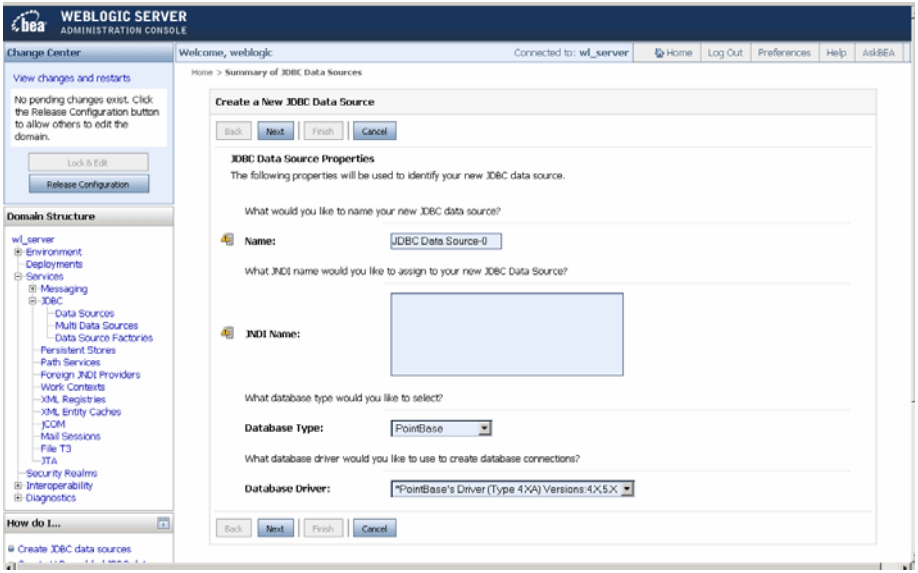
1. Start WebLogic Server for your domain, and log in to the console.
2. In the **Domain Structure** tree, go to **Services > JDBC > Data Sources**. Figure 3-2 shows an example of the **Summary of JDBC Data Sources** page.

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** to view the **JDBC Data Source Properties** page. Figure 3-3 shows an example of the **JDBC Data Source Properties** page.

Figure 3-3 Create New JDBC Data Source in the WebLogic Server Console



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.

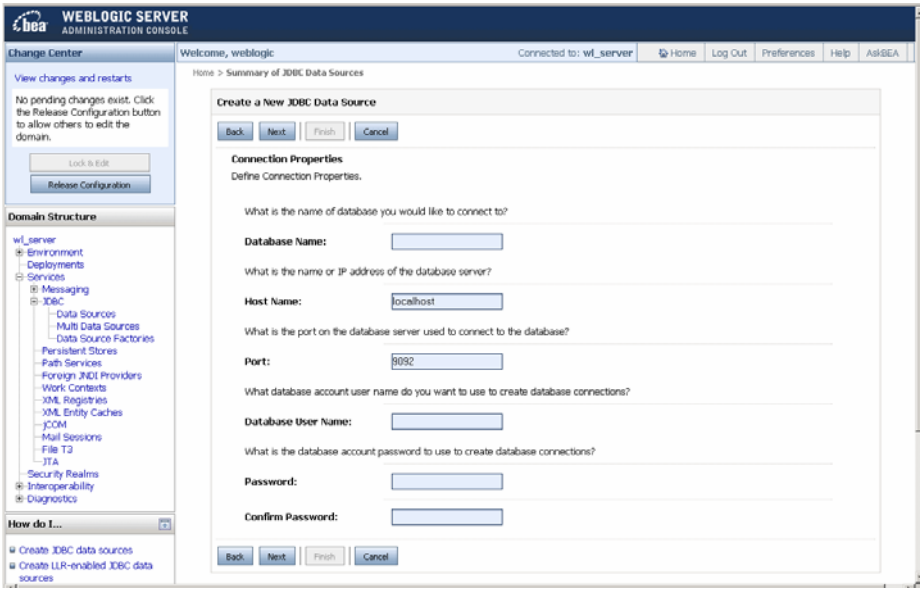
6. Click **Next** to view the **Transaction Options** page.

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

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

8. Click **Next** to view the **Connection Properties** page. Figure 3-4 provides an example of the **JDBC Data Source Connection Properties** page.

Figure 3-4 JDBC Data Source Connection Properties Page in the WebLogic Server Console



9. Within the **Connection Properties** page, use the information in Table 3-6 to complete the dialog fields.

Table 3-6 Connection Properties

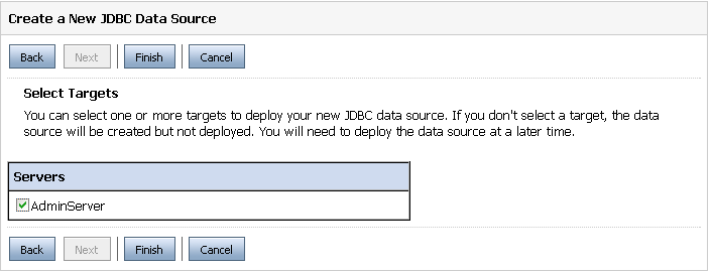
Connection Property	Description
Database Name	The name of the database you are using.
Host Name	Enter the host name used for the database you are using.
Port	Enter the port number of the port hosting your database.
Database User Name	Enter the database user name for the database login required for this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

Table 3-6 Connection Properties

Connection Property	Description
Password	Enter the database password.
Confirm Password	Enter the database password again to confirm.

10. Click **Next** to view the **Test Database Configuration** page.
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



14. Click **Finish**.
15. When finished adding your data source, click **Activate Changes** to update the server.

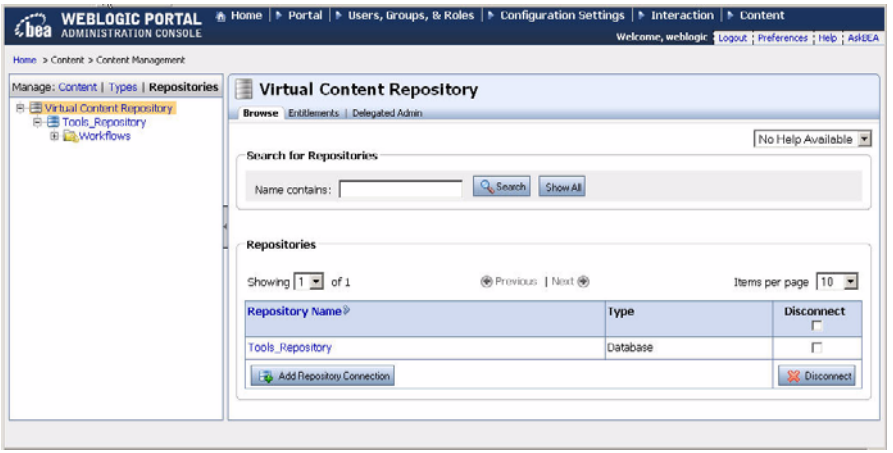
Connecting the New BEA Repository to the Virtual Content Repository

After you have configured the new repository, you need to connect it to the Virtual Content Repository. Use the Portal Administration Console to connect to a new repository.

1. Within the Portal Administration Console, select **Content > Content Management** from the navigation menu at the top of the console.

2. Select **Manage | Repositories**.
3. In the **Manage | Repositories** tree, select the **Virtual Content Repository**.
4. On the Browse tab, click **Add Repository Connection**. Figure 3-6 provides an example of the Browse tab within the Repositories section.

Figure 3-6 Browse Tab within the Manage Repositories Window



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

Table 3-7 Repository Connection Information

Field	Description
Repository Name	The name you give your new repository. For example: `MyNewRepository`
Connection Class	`com.bea.content.spi.internal.ExtendedRepositoryImpl`
Datasource JNDI Name	Use the Datasource JNDI name you created when you connected this repository to your server. The default value of contentDataSource is used for the BEA repository. JNDI names must be unique to the portal domain.
User Name	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	The password field is only used when connecting to a third-party repository. When configuring a BEA repository, you can leave this blank

Table 3-7 Repository Connection Information

Field	Description
Retype Password	Leave this field blank.
Enable Library Services	Unmark this check box if you do not want to use library services with this repository.

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

CHAPTER 4

Using Content Folders in Your BEA Repository

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

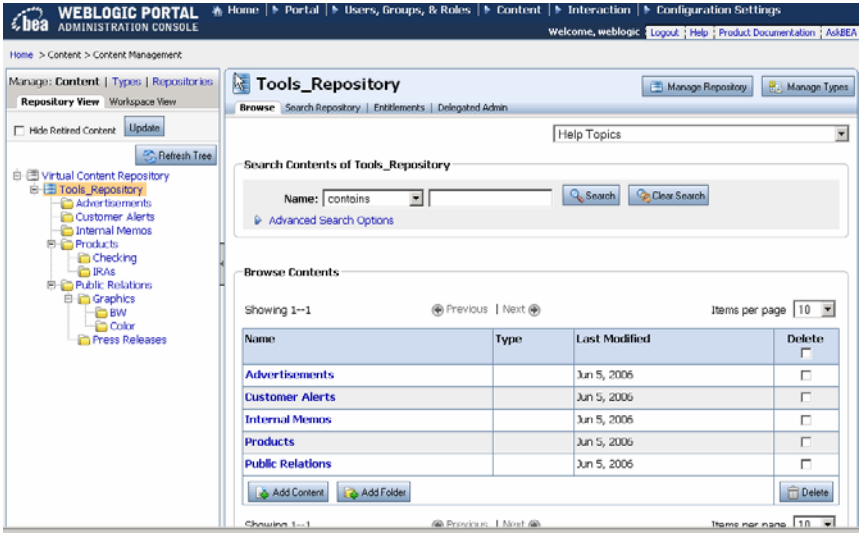
Folders allow you to organize your repository in the following ways:

- Maintain categories for different areas of content. For example, you can create folders per department such as human resources, ad campaigns, and video content.
- Manage your content creation process. When using 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 your content files. You can add administration roles to folders that allow or disallow users to access content folders in your repository, using delegated administration. Visitor entitlements can also be applied at the folder level. For more information about securing content, see Chapter 13, "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.

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
- Changing the Content Workflow for a Folder

Creating a Folder

You can create folders to help organize your content. When you create folders, you can also associate folders with content workflows and/or content types.

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 folder, the content type 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.

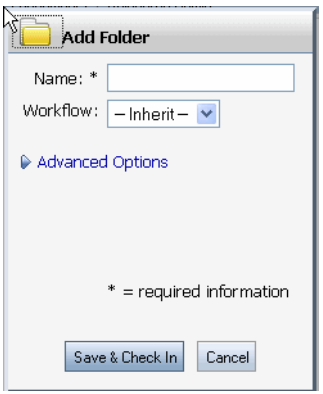
Note: If you are using a filesystem repository, you cannot associate a folder with content type.

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

To create a folder

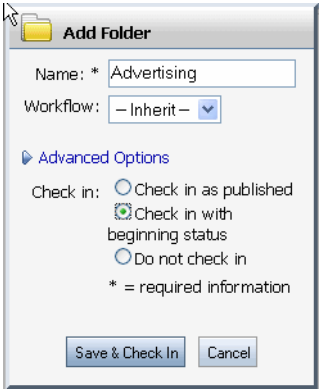
1. Select **Manage | Content** in the resource tree.
2. Select the repository to which you want to add a folder.
3. In the Browse section, click **Add Folder** to view the Add Folder dialog. Table 4-2 shows the Add Folder dialog.

Figure 4-2 Add Folder Dialog



4. In the Add Folder dialog, type a name for your folder.
5. Optionally, associate a content workflow with the folder by selecting one from the Workflow drop-down list.
6. 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



7. Click **Save and 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.

1. Within the repository, select the folder you want to move.
2. Right-click the folder you want to move and select **Move** from the context menu.
3. The **Move Content** dialog displays. Click **OK**.
4. Navigate to the repository location to which you want to move the folder (for example, under an existing folder or at the repository root).
5. Right-click the location and select **Paste** from the context menu.
6. 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, select **Content > Content Management**.
2. In the Manage | Content resource tree, navigate to the content you wish to delete.
3. 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

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

To rename a content item,

1. In the Manage | Content tree, navigate to the content that you want to rename.
2. Right-click the content and select **Rename**.
3. In the Rename dialog, enter in the new name for the content item and click **OK**.

The resource 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. In the main menu, select **Content > Content Management**.
2. View the Manage | Content tree by clicking **Content** at the top of the resource tree.
3. In the Manage | Content tree, navigate to the folder that you want to edit.
4. In the resource tree, click the folder you want to edit to view the Summary page.
5. In the Summary page, click **Workflow** to view the Update Workflow dialog.

6. From the Update Workflow dialog, select a content workflow to use and click **Update**.

The updated workflow information is displayed in the Summary page.

CHAPTER 5

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.

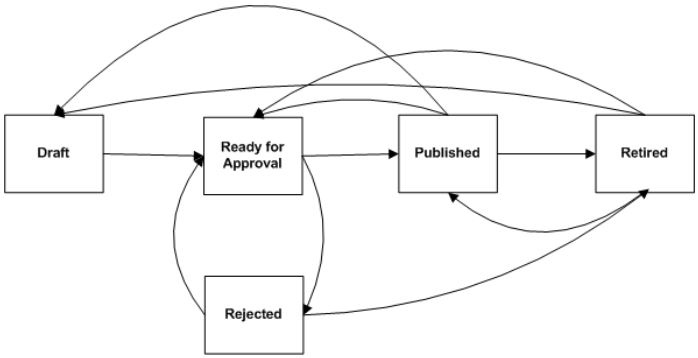
Content workflows are only available in BEA repositories that are library services-enabled. For more information about library services, see Chapter 10, "Adding Content to a BEA Repository."

This chapter includes the following sections:

Using the Default Content Workflow

All content created in a library services-enabled BEA repository uses the default content workflow, unless you implement a customized workflow. WebLogic Portal's 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.

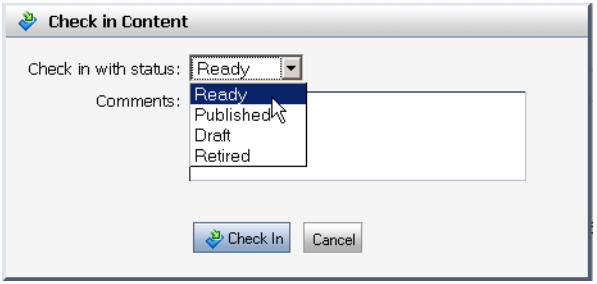
Figure 5-1 Default Content Workflow Diagram



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 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
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

Table 5-1 Default Content Workflow (Continued)

Status	Usage Notes	Required Visitor or Delegated Administration Capabilities
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 WebLogic Portal's 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

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 doesn't 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 Workshop for WebLogic. 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 change a transition, 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 doesn't 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 programatically 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 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 WebLogic Portal's 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>
      <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>`	Indicates the status from which the transition originates. `<from-status>` is a child element of the transition element. **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`. 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).
`<to-status>`	Indicates the ending status of the transition.

Table 5-4 XML Elements of Workflow Document

Workflow XML Element	Usage
`<action-class>`	Indicates what action takes place within this transition. The four default action classes are: • `com.bea.content.virtual.workflow.DraftAction` • `com.bea.content.virtual.workflow.ReadyAction` • `com.bea.content.virtual.workflow.RejectAction` • `com.bea.content.virtual.workflow.PublishAction` • `com.bea.content.virutal.workflow.RetireAction` 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.
`<capabilityConstraint>`	**Delegated Administration:** The syntax for defining a Delegated Administration capability restraint is: • Create: `can_create` • View: `can_view` • Update: `can_update` • Delete: `can_delete` • Reject: `can_reject` • Associate: `can_associate` • Publish: `can_publish` **Visitor Entitlements:** The syntax for defining a Visitor Entitlement capability is: • Create: `can_vis_create` • View: `can_vis_view` • Update: `can_vis_update` • Delete: `can_vis_delete` • Reject: `can_vis_reject` • Associate: `can_vis_associate` • Publish: `can_vis_publish`
`<roleConstraint>`	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.

Table 5-4 XML Elements of Workflow Document

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 Worflow 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:attribute name="id" type="xs:integer"
                              use="required"/>
```

```
            </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 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 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.

This section discusses the following topics:

- Assigning a Content Workflow to a Folder
- Assigning a Content Workflow to a Content Type
- Assigning a Content Workflow to a Content Item

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 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, seethe 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 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, seethe WebLogic Portal Javadoc.

CHAPTER 7

Using WebDAV with Your BEA Repository

WebDAV is a protocol that allows you to communicate with a web server via Internet Explorer and other applications. 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 website.

This chapter includes the following sections:

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

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

- 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 utiltizes memory when adding files to your repository. For this reason, you should add very large files through the Portal Administration Console.

Supported Versions of Microsoft Office

WebLogic Portal's implementation of WebDAV requires Microsoft Office 2000, SP3 or greater, with `MSDAIPP.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 3.

This section discusses the following topics:

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. Select **Manage | Repository** to list the available repositories.
2. In the Manage | Repositories tree, click the respective repository to view the Repository Summary.
3. 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. In the **Manage | Repositories** tree, click the respective repository to view the Repository Summary.
2. Click **Repository Details**.
3. For the `WEBDAV_ENABLED` property, enter a value of false.
4. 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.

This section includes the following topics:

- How WebDAV Determines Which Content Type to Use
- Defining a WebDAV Content Type
- Creating a Content Type That Maps to Microsoft Document Properties

How WebDAV Determines Which Content Type to Use

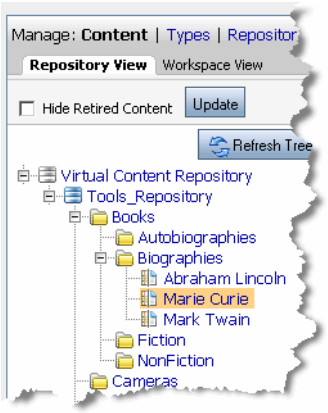
When a user adds a file to your repository using WebDAV, WebDAV associates the content the content type associated with the repository folder where the content is saved.

If no content type is associated with the immediate folder, WebDAV iterates up the repository 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 filesystem repository, you cannot associate content types with folders. Therefore, when using a filesystem 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 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. Select **Manage | Repository** to list the available repositories.
2. In the Manage | Repositories tree, click the respective repository to view the Repository Summary.
3. 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**.

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," "Author," and "Category" with a Microsoft Word document. When these properties are completed for a document using Microsoft Word (**File > Properties**), 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 you can map existing content type property names to the Microsoft property names.

Use the property names listed in Table 7-1 when you define the content type you want to use with Microsoft documents added to your repository by using WebDAV. 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 via 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 via WebDAV. For example, if you add a property value to a Microsoft document and then add document to your repository via WebDAV, you must use the 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 via WebDAV, the property will not be deleted in your repository.

Note: Not all document properties listed in Table 7-1 apply for all Microsoft documents. For example, some properties apply to Microsoft PowerPoint 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
MS0_KEYWORDS	Keywords in the document.	String
MSO_COMMENTS	User comments.	String
MSO_SUBJECT	Subject.	String
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

Table 7-1 Standard Microsoft Document Properties Supported by WebDAV

Name	Description	Data Type
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 PowerPoint 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 the 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”. If you want 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 in order for the mapping to work. For example, if the Microsoft property has a data type of string, the content type property it maps to must also have a string data type.

To map Microsoft document properties to properties within your WebDAV content type, do the following:

1. In the **Manage | Repositories** tree, click the respective repository to view the Repository Summary.
2. 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 (other than what is already provided), you can do so.

For example, within Microsoft Word users can add custom properties to documents using the Custom tab. 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.

You can either add a corresponding property to your WebDAV content type that matches the Microsoft document property name and data type or map the Microsoft property to an existing type, see "Mapping Microsoft Properties to Existing Content Type Properties" on page 7-7

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 programs. The following procedure enables all WebDAV capabilities that are supported by WebLogic Portal.

This section includes the following topics:

- Enabling WebDAV for an Environment
- Adding a Microsoft Word Document to a BEA Repository
- Using Windows Explorer Add a File to the BEA Repository

Enabling WebDAV for an Environment

In order 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. Mark 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 like you would 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 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, you are prompted to log in. Use the user name and password you use to log into the Portal Administration Console.

The file is added to the repository using the default content type and automatically fills in default values and system properties such as the version number, the date and so on, see "Using WebDAV with Your BEA Repository" on page 7-8 for more information.

After adding a file to the repository, log in to the Portal Administration Console and use the Virtual Content Repository to associate additional content properties with the file you just added. For more information about adding content properties, see "Adding Content" on page 9-5.

Using Windows Explorer Add a File to the BEA Repository

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

Note: When adding content to a BEA repository, you need to follow the WebDAV guidelines, see "WebDAV Guidelines" on page 7-2.

CHAPTER 6

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, as well as if the image is color or has alternative text.

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

The BEA repository includes some predefined content types, and you can create your own types to meet your business needs. As a best practice, you should 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

Its 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, and 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 which 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 WebLogic Portal's library services to use content workflows. For more information about library services, see "Working with BEA Repository Content When Using Library Services" on page 9-3.
- **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 6-9.

This section includes the following topics:

- Using Content Type Inheritance
- Using Abstract Content Types
- Using Content Workflows with Content Types

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: If you want 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 "Using Content Folders in Your BEA Repository" on page 4-1.

For example, you can create a content type that should be used 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.

For example, 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 so on.

With content type inheritance, each time you edit a parent content type, the children content type are 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 that you can add to content types. Abstract content types can only be used within the context of an existing content type. By using abstract content types, you can ensure consistency across all content types that include that abstract content type.

For example, if you want to include address information in multiple content types, you can create an abstract content type called “address” and then next that content type within another content type. When you create an abstract content type, you cannot create content based on that type alone. The only way an abstract content type can be used is within another content type, by using a nested content type or with inheritance. For more information about nested content types, see “Using Nested Content Type Properties” on page 6-9.

Abstract content types cannot be associated with content, unless used as part of another content type. If you mark a content type as “abstract”, you prevent content contributors from creating content based on that content type. For more information about creating abstract content types, see “Marking a Content Type as 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 BEA’s Library Services” on page 9-4.

Working with Content Types

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

Note: You can allow or disallow 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

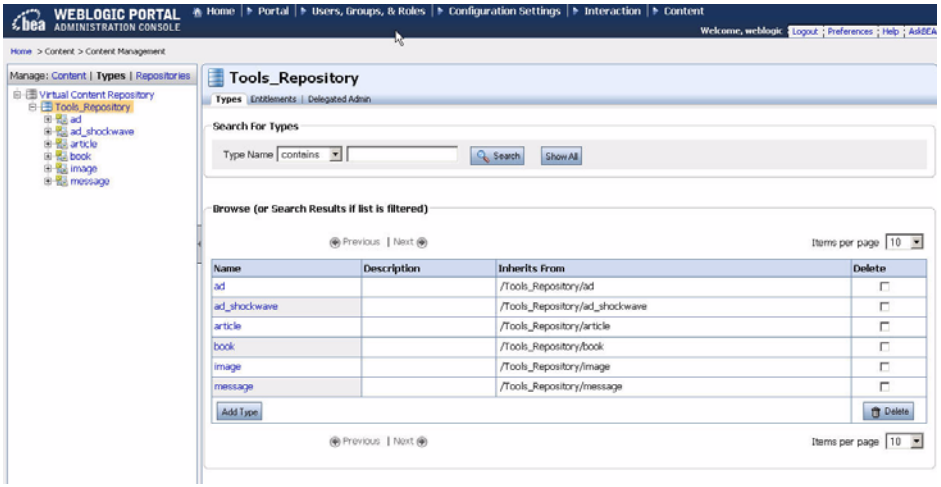
Content types are created within the Portal Administration Console.

To create a new content type:

1. Within the Administration Console, select **Content > Content Management**.

2. Select **Manage | Types** to view the Types resource tree.
3. 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 the repository area, click **Add Type**. The Add Type dialog displays as shown in Figure 6-2.

Figure 6-2 Add Type Dialog

5. In the Add Type dialog, enter a name and description for your content type.

6. Optionally, you can 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, you can 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 Types tab displays. You can specify property definitions for this content type using the Types tab.

Marking a Content Type as Abstract

Optionally, you can mark a content type as 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 mark a content type as abstract:

1. Select **Types** within the resource tree to view the Manage | Types resource tree.
2. Select the repository in which your content type resides.
3. Click the content type you want to make abstract. The Summary tab displays in the right pane.
4. In the Summary tab, click **Name & Status** to view the Edit dialog.
5. In the Edit dialog, mark the **Is Abstract** check box.
6. 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 Portal Administration Console, select **Content > Content Management**.
2. View the Manage | Types tree by clicking Types at the top of the resource tree.
3. In the Manage | Types tree, navigate to the content type you wish to delete.
4. Right-click the content type and select **Delete** from the pop-up menu.
5. A pop-up window displays asking you to confirm that you want to delete the content type. If you want to proceed, 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 can be 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.

The properties you associate with image files also determine how the file can be used in your portal. For example, you can 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 6-12.

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

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

This section discusses the following topics:

- Supported Data Types
- Using Nested Content Type Properties
- Property Options

- Using Primary Properties
- Setting Property Choice Lists and Default Property Values
- Using Link Properties
- Defining Content Properties for Interaction Management
- Define the Properties of a Content Type

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 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.

Table 6-1 Supported Data Types (Continued)

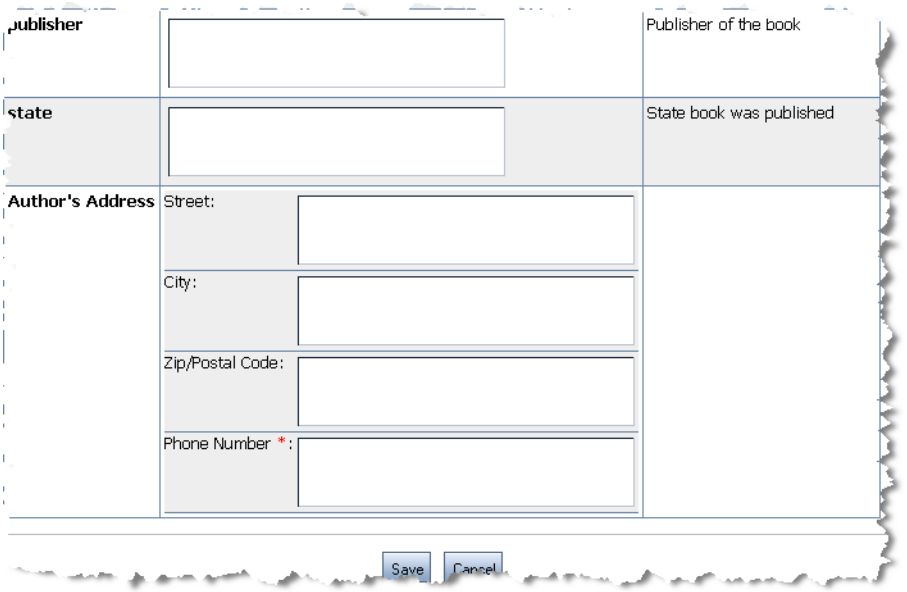
Data Type	Value Definition
Nested Content Type	A data type that contains another content type. Nested Content Type properties cannot be marked as primary. For more information about nested content types, see "Using Nested Content Type Properties" on page 6-9.
Link	A data type that contains a link to 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 you nest a content type as a property, the properties of that nested type are grouped together to make up the value of the nested property. values for that 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 you need the address 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 6-14.

Figure 6-3 Example of a Nested Content Type Property



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, it's value cannot be edited.
Primary Property	Marks this property as the primary property for this content type. Primary properties can be used within content searches and placeholders to optimize queries. For example, you can retrieve a list of content according to the primary property of it's content type. A content type can have only one primary property.

Table 6-2 Available Property Options

Property Option	Usage
Searchable	Makes this property accessible through developer content queries. Occasionally, you will not want to make properties searchable. For example, salary data might be confidential and not searchable.
Is Explicit	Allows you to map this property value directly to a value contained in the content management database where your content is stored. When you mark **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*.

Using Primary Properties

Primary properties are used by portal developers 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.

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.

You can set choice lists and/or include default values when you add a property to a content type, see "Define the Properties of a Content Type" on page 6-14.

Using Link Properties

You can create properties that allow content contributors to associate content items. Content contributors can link to content within the same repository or another repository 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 can be 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. See "Define the Properties of a Content Type" on page 6-14 for detailed instructions on adding a link property.

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 personalization, 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 to run a "campaign" telling employees that they can update their benefit information. In the internal human resources portal, you can create a campaign that runs from November 1 to 30, during which time the campaign displays an Open Enrollment graphic in the portal header region and, when employees make changes to their benefits.

Adding specific properties to your content types can ensure that developers can take full advantage of the interaction management tools that are available. The following properties should be used for content with which you want to retrieve with interaction management activities.

Tip: You can use an abstract content type to add interaction management properties to multiple content types within your repository, see "Using Abstract Content Types" on page 6-3. To do this, create a content type that includes each interaction management property and then mark it as Abstract. You can then add this abstract type to any content type within your repository.

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 as described in Table 6-3.

Table 6-3 Interaction Management Properties for Content

Content Property	Type	Description
adTargetUrl	String	Makes an image clickable and provides a target for the clickthrough, expressed as a URL. The Event Service records the clickthrough. Depending on how you want to identify the destination of the ad clickthrough, use **adTargetUrl**, **adTargetContent**, or **adMapName**.
adTargetContent	String	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. You can view a content item's unique ID by selecting the content item in the WebLogic Portal Administration Console and viewing the description in the Edit Content window. Depending on how you want to identify the destination of the ad clickthrough, use **adTargetUrl**, **adTargetContent**, or **adMapName**.
adMapName	String	Makes an image clickable, using an image map to specify one or more targets. The value for this attribute is used in two locations: • In the anchor tag that makes the image clickable: `<a href=value> <img> </a>` • In the map definition: `<map name=value>` If you specify a value for **adMapName**, you must also specify a value for **adMap**. Depending on how you want to identify the destination of the ad clickthrough, use **adTargetUrl**, **adTargetContent**, or **adMapName**.
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 `newwindow`.
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 `Close this window` 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.

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

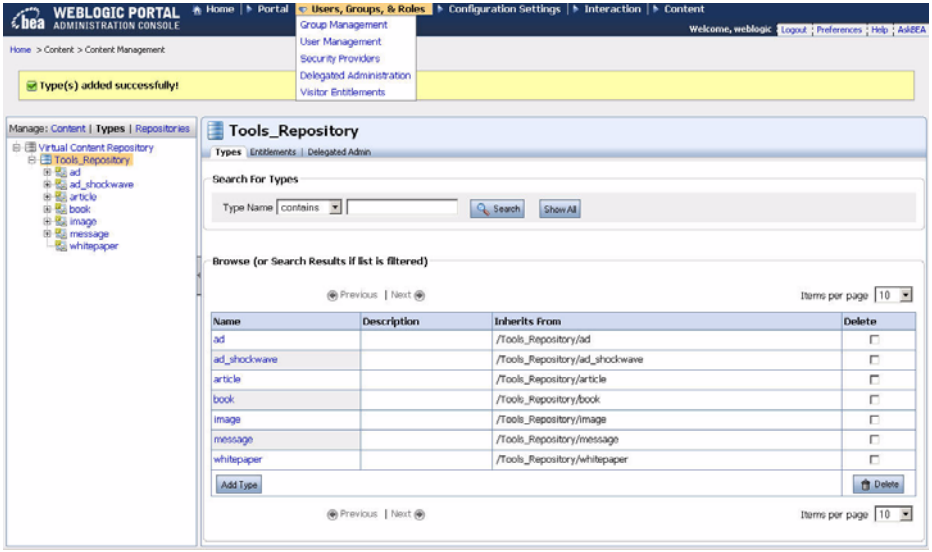
Content Property	Type	Description
adAltText	String	Specifies a text string for the **alt** attribute of the `<img>` tag. If you do not include this attribute, the `<img>` tag does not specify an **alt** attribute.
adBorder	Integer	Specifies the value for the border attribute of the `<img>` tag. If you do not include this attribute, the border attribute is given a value of **0**.

Define the Properties of a Content Type

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

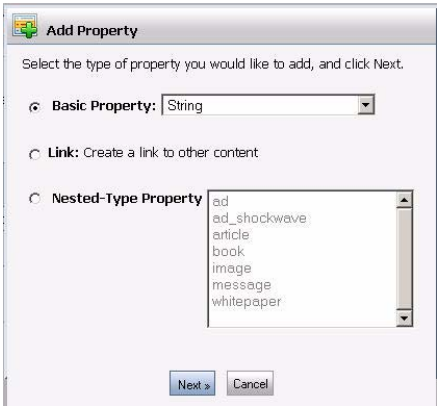
1. From the Types tab, select the type to which you want to define properties. Figure 6-4 shows an example of the Types tab.

Figure 6-4 The Types Tab within the Manage Types Window



2. Click the Properties tab.
3. Click **Add Property**. The Add Property dialog appears as shown in Figure 6-5.

Figure 6-5 Add Property Dialog

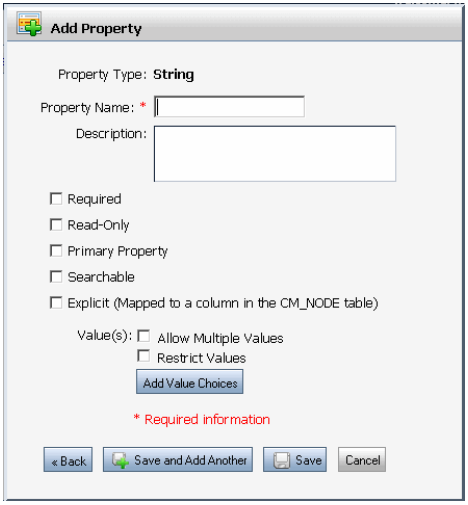


4. In the Add Property dialog, select the type of property you want to add: **Basic Property**, **Link** or **Nested-Type Property**.
   - To add a **Basic Property,** select a data type from the drop-down list. See "Supported Data Types" on page 6-8 for more information about data types.
   - To add a **Nested-Type Property**, select a content type from the provided list. A nested-type property will add the property definitions from the selected type to the type you are creating.

   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.

   - To add a **Link** property, select **Link**.
5. Click **Next** to view the Add Property dialog. Figure 6-6 shows an example of the Add Property dialog.

Figure 6-6 Add Property Dialog



6. In the Add Property dialog, type a name for the property you are creating.
7. Mark the property option you want to assign to this property. For more information about property options, see "Property Options" on page 6-10.
8. In the Add Property dialog, mark **Allow Multiple Values** if you want this property to hold more than one value.
9. Mark **Restrict Values** if you want users to select from a list of values that you provide. This means they can only enter a value which is included on the list you provide.
10. To provide a default value, click **Add Value Choices** to view the Add Value Choices dialog.
    a. In the Value Choices dialog, enter a value in the **Value** field.
    b. Click **Save**.
11. To provide a list of values, ensure that Allow Multiple Values is marked. Then click **Add Value Choices** to view the Add Value Choices dialog.
    a. In the Value Choices dialog, enter a value in the **Value** field.
    b. Optionally, mark the **Default** check box to mark this as a default value.
    c. Click **Add Another Value** and continue adding values until finished.
    d. Click **Save**.

12. When finished, select either **Save** to exit the Add Property dialog or **Save and Add Another** to use the Add Property dialog to add another property.

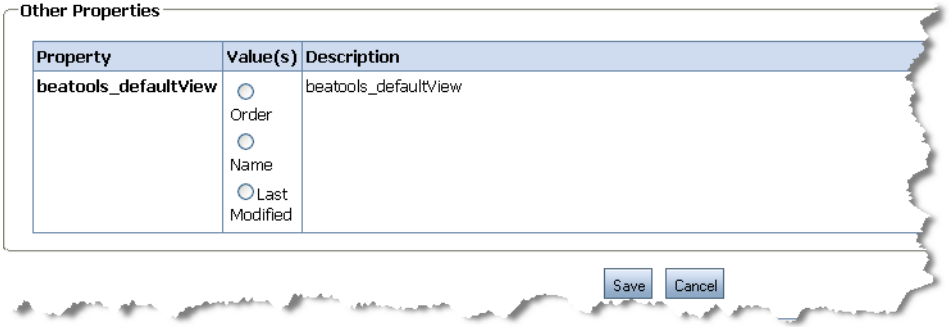
Re-ordering Content Within A Folder Using Properties

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. In the Manage | Content tree, select the folder that you want to modify.
2. Click the **Summary** tab.
3. If using library services, check out the folder by clicking **Check Out** in the Versioning and Workflow section.
4. Click **Properties** to modify existing property values.
5. For the *beatools_defaultview* property, select the **Edit** check box and click **Edit**.
6. In the Other Properties section, select a new value for the *beatools_defaultview* as shown in Figure 6-7.
   - **Order** allows users to use a custom order, see "Re-Ordering Content" on page 9-9.
   - **Name** sorts the content in alphabetical order by name
   - **Last Modified** sorts the content by the date it was last modified

Figure 6-7 Available Property Values for the beatools_defaultview Property



7. Click Save when finished making your selection.
8. If using library services, check in your changes.

Out-of-the-Box Content Types

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

Although these 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*.

This section includes a reference of the content types that are included and the property definitions they use. 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 can be used for content that is designed to be an advertisement. It includes properties that open and close popup windows as well as presents alternative text when the portal user's browser does not support viewing the content.Table 6-4 provides information about the properties includes 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.
adTargetURL	String	The target URL for image clicks.

Table 6-4 Ad Content Type Properties (Continued)

Property Name	Data Type	Description
adWinTarget	String	If set, target render will be in a popup window with this name.
adWinClose	String	If set, and adWinTarget set, a close link will be included.
adWinTitle	String	If adWinTarget set, this will be 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 for when to display the article in your portal. These dates can be used when configuring campaigns. Table 6-5 details the properties includes 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

The book content type can be used 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 includes 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

You can use the message content type to store message content such as alerts. Table 6-8 provides information about the properties includes 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 to associate with folders, you must have those content types inherit from the CM_FOLDER_BADGE content type. For more information about type inheritance, see "Using Content Type Inheritance" on page 6-3.

CHAPTER 8

Connecting to a Third-Party Repository

WebLogic Portal's 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 via BEA's JSR 170 Connector, you can use BEA's JSR 170 Connector, see "Working with a JSR 170-Compatible Repository" on page 8-8. For more information about JSR170, see the JSR 170 website.

If you are using a third-party repository from a vender that has not written an implementation for WebLogic Portal's Virtual Content Repository, you can write your own using BEA's Service Provider Interface (SPI).

This chapter includes the following sections:

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. If you are connecting to a JSR 170-compatibly repository, you do not need to write an SPI implementation.
2. Adding the repository SPI jar(s) 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 repository to the Virtual Content Repository using the Portal Administration Console. Depending on your implementation, you need to set various repository properties for your third-party repository so it connects with the Virtual Content Repository. These repository properties are SPI-specific and should be described in the SPI documentation.

Creating an SPI Implementation

The SPI implementation runs inside a WebLogic Portal enterprise application. Different implementations may affect scalability and performance for WebLogic Portal.

The following section gives a brief overview of some interfaces a SPI implementation for accessing a third-party repository must implement, and some of the classes the SPI implementation can use. See the Content SPI JavaDoc for detailed information.

The SPI includes two types of interfaces:

- Connection interfaces which operate as the gateway to the implementation and are used to conduct repository tasks such as connecting to the repository and so on.
- Service interfaces which conduct typical content management tasks such as adding, updating, deleting, and accessing content.

The SPI also includes classes that allow you to use error messages (repository exceptions) to indicate if operations are not allowed in the repository. For example, the SPI allows for content to be added to folders, if the repository does not allow that behavior then it must throw a `RepositoryException` stating so.

If a method is not implemented at all, then a `UnsupportedRepositoryOperationException` can be thrown.

This section includes the following topics:

Repository Connection Interfaces

Repository Connection Interfaces include the interfaces listed in Table 8-1. In order for a client of the SPI to connect to the services, the repository connection interfaces must be implemented. These interfaces allow an SPI implementation to be plugged into the BEA content management framework.

For more information about these repository connection interfaces, see the WebLogic Portal JavaDoc for the com.bea.content.spi package.

Table 8-1 Repository Connection Interfaces

Service Interface	Usage Notes
`Repository`	`Repository` defines the configuration for a content repository and also provides access to the content repository services through the `Ticket`.
`ExtendedRepository`	`ExtendedRepository` provides access to repository connections
`Ticket`	`Ticket` is the interface given back to the client after calling connect to a Repository with `Credentials` that are authenticated. It is the gateway to the content repository services.
`ExtendedTicket`	`ExtendedTicket` enables repository services that are available in WebLogic 9.2. For example, content workflows can be used. For more information about content workflows, see "Using Content Workflows in Your BEA Repository" on page 5-1.
`Credentials`	`Credentials` defines the credentials for a user attempting access to the content repository. Both authentication and authorization can be based on the `Credentials`.

Service Interfaces

Service Interfaces are used to perform the primary content management tasks within your repository such as adding content, updating it and deleting it.

Service interfaces are expected to be transactional (where necessary), threadsafe and scalable. There can be one implementation for all interfaces, or a separate implementation for each service interface. All service methods throw a general `com.bea.content.RepositoryException`. The SPI also includes a few standard exceptions that extend the `RepositoryException`.

Table 8-2 lists the service interfaces available with the content SPI. For more information about these service interfaces, see the WebLogic Portal JavaDoc for the com.bea.content.spi package.

Table 8-2 Service Interfaces

Service Interface	Usage Notes
`NodeOps`	`NodeOps` is the service interface for performing operations on content and its properties. This includes accessing, creating, updating, deleting, copying, moving and renaming content and content folders.
`ExtendedNodeOps`	`ExtendedNodeOps` provides access to newer features that are included in WebLogic Portal 9.2 such as dynamically sorting and filtering query results.
`ObjectClassOps`	`ObjectClassOps` is the service interface for performing operations on `ObjectClasses` (content types) and their `PropertyDefinitions` and `PropertyChoices`. This includes creating, updating and retrieving `ObjectClasses`.
`ExtendedObjectClassOps`	`ExtendedObjectClassOps` provides access to newer features for objectClasses (content types) that are included in WebLogic Portal 9.2 See "Using Content Types in Your BEA Repository" on page 6-1 for more information.
`SearchOps`	`SearchOps` is the service interface for performing content searches. The search is based on the Search object defined in `com.bea.content.expression` along with the expression structure in the `com.bea.p13n.expression` package. Search is for any content that matches the property and content type used in the expression. There are system properties that can be part of the search and are defined in the Search class.
`ExtendedSearchOps`	`ExtendedSearchOps` is used for full-text search content indexing operations.

Example of a Simple SPI Implementation

Listing 8-1 is an example of a repository implementation is that is called by the Virtual ContentManager.

Listing 8-1 Example of Repository Implementation

```
public class RepositoryImpl implements Repository
{
  Properties properties;
  String name;

  public Ticket connect(Credentials credentials)
  {
    return new TicketImpl(credentials, properties);
  }

  public Ticket connect(String userName, String password)
  {
    Subject subject = com.bea.p13n.security.Authentication.
    getCurrentSubject();
    Credentials credentials = new Credentials(subject);
    return new TicketImpl(credentials, properties);
  }
  public Properties getProperties()
  {
    return properties;
  }
  public void setProperties(Properties properties)
  {
    this.properties = properties;
  }

  public String getName()
  {
    return name;
  }
```

```
  public void setName(String name)
  {
    this.name = name;
  }
}
```

Connecting to a Third-Party Repository

Once you have an SPI implementation, you need to connect the third-party repository to the Virtual Content Repository using the Portal Administration Console. After 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 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.

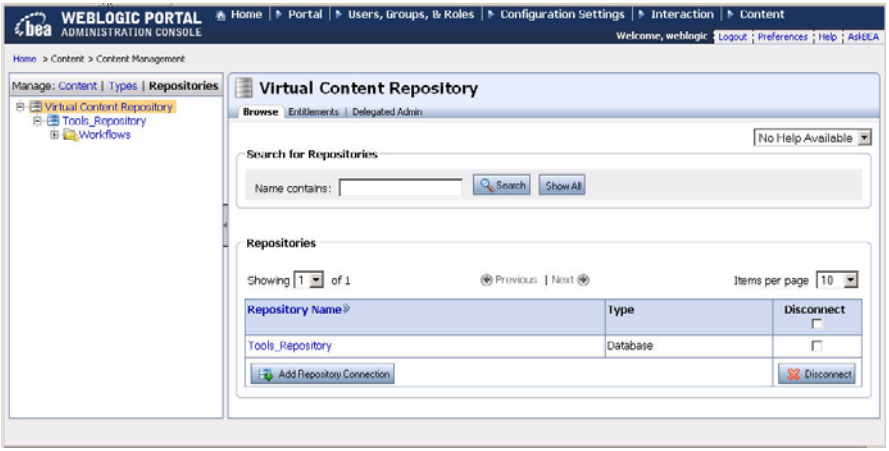
Note: BEA's library services are not supported for use with SPI implementations.

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 you need to configure and the connection class you should use.

To connect a repository to the Virtual Content Repository:

1. Within the Portal Administration Console, select **Content > Content Management** from the navigation menu at the top of the console.
2. Select **Manage | Repositories**.
3. In the Manage | Repositories tree, select the **Virtual Content Repository**.
4. On the Browse tab, click **Add Repository Connection**. Figure 8-2 provides an example of the Browse tab within the Repositories section.

Figure 8-1 Browse Tab within the Manage Repositories Window



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

Table 8-3 Repository Connection Information

Field	Description
Repository Name	The name you give your new repository. For example: `MyNewRepository`
Connection Class	Use the SPI connection class you have created or that has been provided by your third-party vendor. Be sure you have added this class to your classpath.
Datasource JNDI Name	Used only for BEA repositories.
Username	If the SPI implementation requires a username, enter it here. When configuring a BEA repository, you can leave this blank.
Password	If the SPI implementation requires a password, enter it here. When configuring a BEA repository, you can leave this blank
Retype Password	If you entered a password, re-enter it here.
Enable Library Services	Unmark this check box if you do not want to use library services with this repository. Library services can ONLY be used with BEA repositories.

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

Working with a JSR 170-Compatible Repository

WebLogic Portal provides a connector to repositories which implement the JSR 170 Specification such as Day Software's CRX. If you want to access a repository via 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*.

This section discusses the following topics:

- Searching within a JSR 170 Repository
- Connecting to a JSR 170 Repository

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 (system properties, MIME types, and so on) are supported by the JSR 170 specification, so 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:

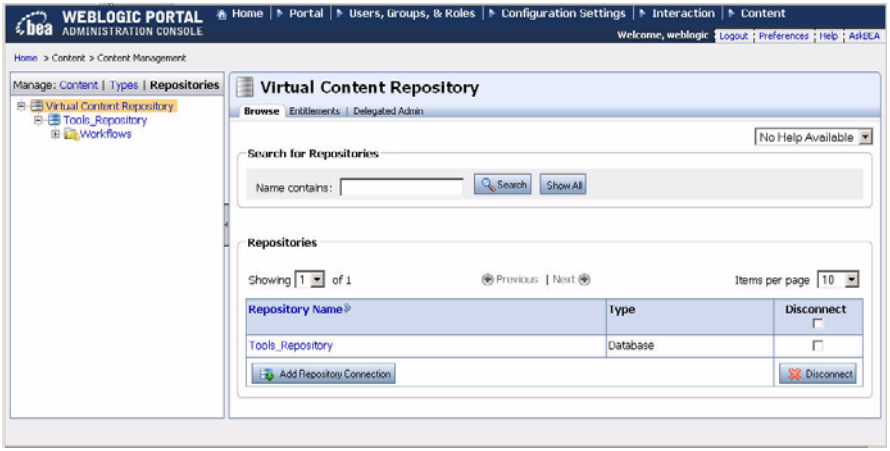
- containsall
- likeignorecase

Connecting to a JSR 170 Repository

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

1. Select **Manage | Repositories** to view the Repositories tree.
2. Click the **Virtual Content Repository**.
3. In the Browse tab, click **Add Repository Connection**. Figure 8-2 provides an example of the Browse tab within the Repositories section.

Figure 8-2 Browse Tab within the Manage Repositories Window



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

Table 8-4 Repository Connection Information

Field	Description
Repository Name	The name you give your new repository. For example: `MyNewRepository`
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: `com.day.content.spi.jsr170.JNDIRepository`
Username	Enter the username required to connect to you third-party repository the third-party repository you are using.
Password	Enter the password required to connect to your third-party repository.
Retype Password	Re-enter the password required to connect to your third-party repository.
Enable Library Services	Unmark this check box to ensure library services are disabled. **Note:** Library services are not supported for third-party repositories.

5. Click **Save**.
6. Within the Repositories section, click the repository you just created to verify that it has been created and view its Repository Summary.
7. Add the properties in Table 8-5 to your repository.

Table 8-5 JSR170 Connector Properties

Repository Property	Required Value	What it does:
`jsr170.hide.builtin`	true	This property prevents repository-specific types and properties from interfering with the Virtual Content Repository.
`jsr170.uid.mapping`	uuid	This property maps the IDs generated by your repository to the internal IDs needed by the Virtual Content Repository.

Repeat the following steps for each property you want to add:

a. In the Properties section, click **Add Property**.

b. In the Name field, enter the name of the property you want to add.

c. In the Value field, enter the value of the property.

d. 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.

Part II Development

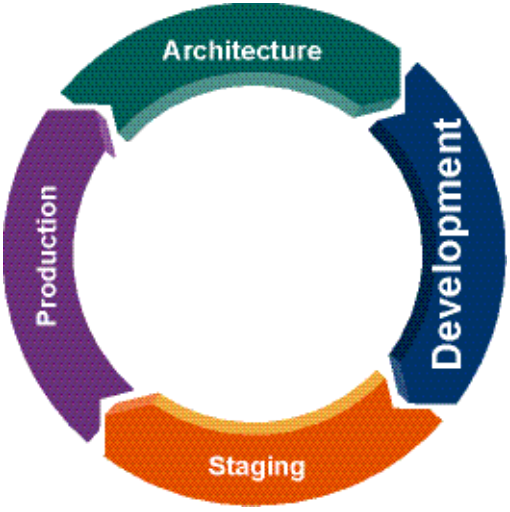
Within content management, the development phase covers both creation of content as well as delivering that content to your portal users. Using the Portal Administration Console, content contributors add content to the content repository using workflows and versioning. Using Workshop for WebLogic, developers 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 chapter:

- Chapter 9, "Adding Content to a BEA Repository"

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 Figure 1-1:

Figure 1-1 Portal Life Cycle



CHAPTER 9

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 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
- Using Content Workflows

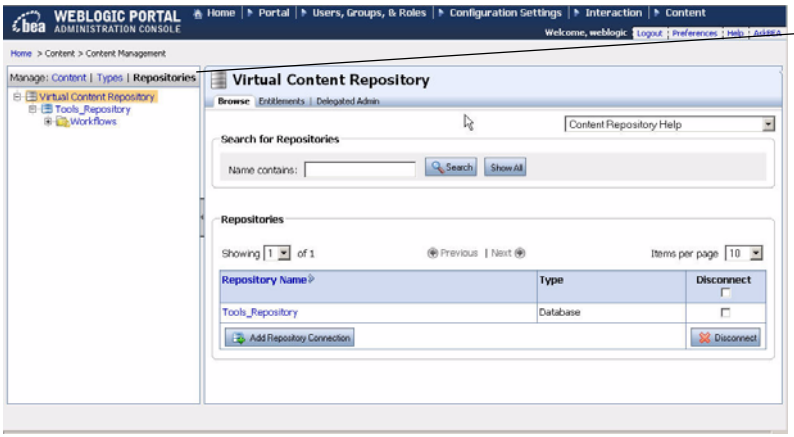
Note: Before content can be created, you must first create content types, design your repository hierarchy, and (if using Library Services) add content workflows, see Chapter 4, "Using Content Folders in Your BEA Repository",Chapter 5, "Using Content Workflows in Your BEA Repository" and Chapter 6, "Using 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 Portal Administration Console, see Figure 9-1.

Figure 9-1 The Virtual Content Repository within the WebLogic Portal Administration Console



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

Table 9-1 lists each view and the tasks you accomplish in each view.

Table 9-1 Summary of the Views of the Virtual Content Repository

Virtual Content Repository View	Associated Tasks
Content view (Click **Manage | Content**)	• Add content to the repository • Search for content • Delete content • Modify content If using BEA's library services, you can also: • View version history • Check out content • Check in content
Types view (Click **Manage | Types**)	• Add content types • Modify content types (adding or removing property definitions) • Delete content types
Repository view (Click **Manage | Repositories**)	• Connect repositories to the Virtual Content Repository • Edit repository properties such as search settings and caches If using BEA's library services, you can also: • 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 BEA's library services which enables 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. Library services also 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 disallow 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*.

You can add content to a repository using content management tools in the Portal Administration Console.

Note: You can also add content to your repository using Microsoft Explorer or other Microsoft applications, if your administrator has enabled your repository to do so. For information about adding content to your repository using Microsoft Explorer or other applications, "Using WebDAV with Your BEA Repository" on page 7-8.

This section includes the following topics:

Overview of BEA's Library Services

When you use a BEA repository to store your content, you can enable that repository to use library services. Library services provide versioning, workflow, as well as customized workspaces for content contributors. For instructions on how to enable a BEA repository to use library services, see "Enabling Library Services for a BEA Repository" on page 3-3.

- Workflow management that provides a customizable workflow for developing content

- Content workspaces that provides a simplified user-based views of content that is currently in progress
- Version control of content items and content types

Table 9-2 Overview of BEA Repository Library Services

Library Service	What it provides:
Content Workflow	The Content Workflow allows you to move content through a set of statuses that can be further managed through WebLogic Portal's Delegated Administration. WebLogic Portal's default content workflow includes the following statuses: Draft, Ready for Review, Rejected, Published and Retired.
Content Workspace	The Content Workspace 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. Use the Content Workspace to: • View items that are currently assigned to your role. All users within a particular role view/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	Content Versioning allows you to keep track of multiple versions of a content item. Versioning provides a more powerful alternative to keeping backup files. With content versioning, you can easily view previous versions of a content item if needed, as well as 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 the file with a content type or you can create HTML content directly in the repository. To create HTML content, see "Creating HTML Content" on page 9-7.

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

Note: This example also 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. In the Manage | Content tree, select the repository to which you want to add content.
2. In the Browse area, click **Add Content** to view the Add Content dialog.
3. In the Add Content dialog,
   a. Type a name for the content.
   b. Select the book content type from the Type drop-down list.
   c. Click **Add** to view the Add Node page.
4. In the Add Node page,
   a. Associate content with a file by clicking **Upload File** in the Primary Properties area.
   b. Click **Browse** to select a content file from your filesystem.
   c. If you know the type of encoding you wish to use, select it from the drop-down list. If not, leave the default.
   d. Optionally, select the MIME format that matches the content you selected.
   e. In the Other Properties area, provide values for the properties listed. These property values are used by portal developers to retrieve your content.
5. Click **Save**. Your content item is saved and remains checked out of the repository.

   While content is checked out of the repository, it is displayed in the Workspace View under the Checked Out Items folder.
6. To view your content item, select the Workspace View tab and navigate to your content item within the Checked Out Items folder.
7. Click the content item you created to view its properties.
8. In the Versioning and Workflow area, click **Check In** to view the Content Check In dialog.
9. Select a status from the Check in with Status drop-down list.
10. Enter any comments in the comments text box.

11. Click **Check In**.

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 HTML file. The content editor allows you to create HTML forms as well as HTML documents.

To create HTML content:

1. In the main menu, select **Content > Content Management**.
2. View the Manage | Content tree by clicking **Content** at the top of the resource tree.
3. In the Manage | Content tree, navigate to the repository location where you want to create the content item.
4. In the Browse tab, click **Add Content**.
5. In the Name field, enter the name for the new content item.
6. In the Type field, select the content type you want to associate with the content item.
7. In the Add Node tab, enter the required property values.
8. To add an HTML file using the content editor, use the following steps:
   a. In the Primary Property section, click **Create Document**.
   b. Use the content editor to add an HTML file.
9. 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 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. In the main menu, select **Content > Content Management**.
2. View the Manage | Content tree by clicking **Content** at the top of the resource tree.
3. In the Manage | Content tree, navigate to the content that you want to edit. If you are using library services, content will be in the Workspace View > My Assigned Items folder or in the Repository View.

   Note: If you are NOT using library services, you can view content by choosing **Manage | Content** in the resource tree and navigating to the content you want to modify.
4. In the resource tree, click the content you want to edit to view the Summary page
5. In the Versioning and Workflow section of the Summary page, click **Check Out**.
6. Navigate to the Checked-Out Items folder in the Workspace View and click 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** to view the Edit Name dialog.
   b. In the Edit Name dialog, type the new name and click **Update.**
   c. If finished modifying your content, click **Check In**.

      Note: If you are NOT using library services, you do not need to check in content.
   d. Optionally, in the Check In dialog, choose a new workflow status for the content.
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 marking the check box in the Edit column.
   c. Click **Edit** to view the value text box for each property you selected.
   d. Edit the value and click **Save**.
   e. If finished modifying your content, click **Check In**.

Note: If you are NOT using library services, you do not need to check in content.

f. Optionally, in the Check In dialog, choose a new workflow status for the content.

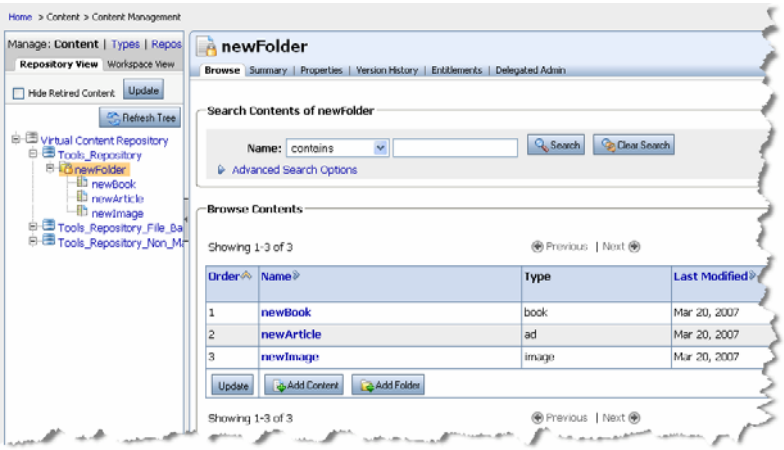
Re-Ordering Content

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

To re-order content within a folder:

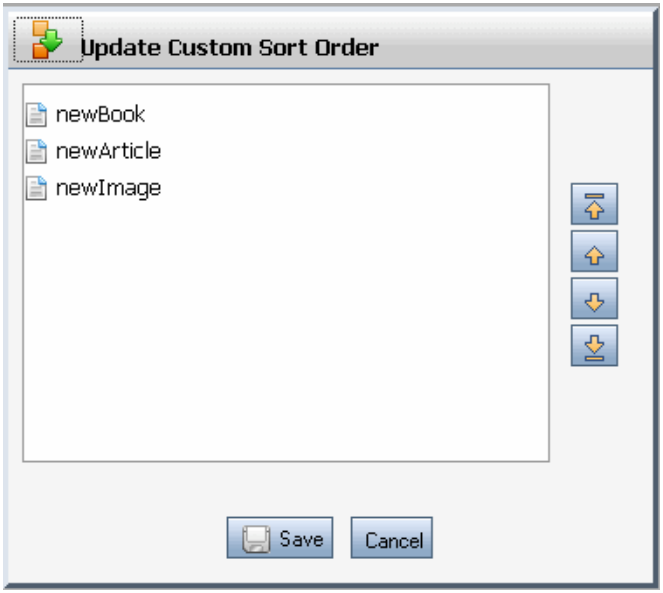
1. Select the folder to which you want to order content.
2. In the Browse Contents section of the Browse tab, click **Order** as shown in Figure 9-2.

Figure 9-2 Re-ordering Content in the Browse Contents Tab



3. Click Update to view the Update Custom Order dialog box as shown in Figure

Figure 9-3 Update Custom Order Dialog Box



4. Use the navigational arrows in the Update Custom Order dialog box to re-order your content and click **Save** when finished.

Tip: It is also possible to change the default sort order that is used by modifying the content type properties of the folder. For more information, see "Re-ordering Content Within A Folder Using Properties" on page 6-17.

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. In the Properties section, select the property you want to edit by marking the check box in the Edit column. If editing a primary property, click **Edit** in the Edit column.
2. In the Property section, click **Download File**.
3. In the Open dialog, choose to save to disk or to open the file.

4. After copying the file to your local drive, make and save your changes.
5. When ready to upload your changes, click **Upload File**.
6. Click **Browse** to navigate to your updated file.
7. Click **Save**.
8. If finished modifying your content, click **Check In**.

   Note: If you are NOT using library services, you do not need to check in content.

9. To change the workflow associated with the content:

   a. Within the Summary page, click **Versioning & Workflow** to view the Update Workflow dialog.

   b. In the Update Workflow dialog, select a new workflow from the drop-down list.

   c. Click **Update**.

   d. If finished modifying your content, click **Check In**.

Note: If you are NOT using library services, you do not need to check in 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 9-12.

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 version, you can choose to retire a content version instead, see "Retiring Content" on page 9-20 for more information.

To delete content:

1. From the main menu, select **Content > Content Management**.
2. In the Manage | Content resource tree, navigate to the content you wish to delete.
3. Right-click the content item you want to delete and select **Delete** from the menu.
4. In the Delete dialog, click **Delete**.

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. In the Manage | Content tree, navigate to the content that you want to edit.
2. Right-click the content and choose **Move**.
3. Click **OK** in the Move dialog.
4. Navigate to the repository location (folder or content) where you want to move the content.
5. Right-click on the repository location (folder or content) where you want to copy the content item and choose **Paste**.
6. Click **OK** in the Paste dialog to confirm.

Linking Content

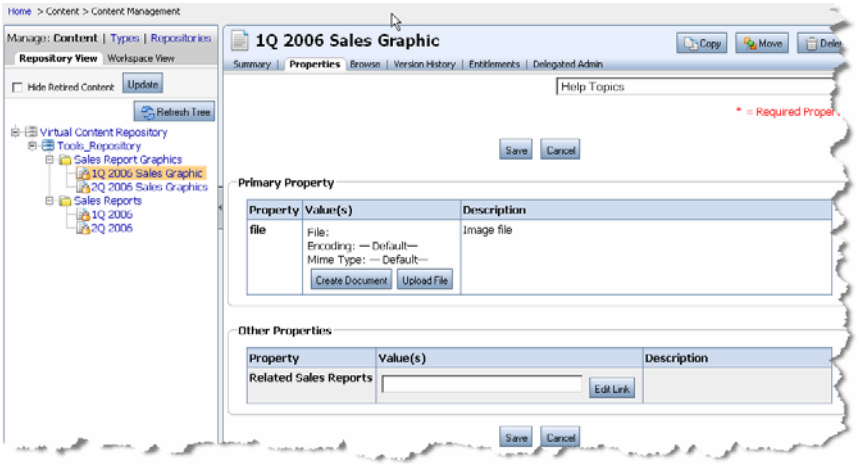
Some content types allow you to link to other content within the Virtual Content Repository. Content links allow you to associated content so it can be easily maintained and updated.

In order 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.

1. Navigate to the content that you want to add a content link.
2. Click the **Properties** tab.
3. In the Other Properties section, select the property you want to edit by marking the check box in the Edit column.
4. Click **Edit**.
5. Click **Update Link** to view the Update Link dialog. Figure 9-4 shows an example of a link property.

Figure 9-4 Example of a Link Property



6. Use the Repository drop-down list to select a repository to search.
7. Use the search text box to find the content to which you want to link. Type a keyword in the text box and click **Search**.
8. Within the search results, click the content to which you want to link.
9. Click **Update Link**.

When finished, you can view the path to the linked content using the Properties tab.

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. In the **Manage | Content** tree, navigate to the content that you want to rename.
2. Right-click the content and select **Rename**.
3. In the **Rename** dialog, enter in the new name for the content item and click **OK**.

The resource 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 9-12.

To copy content or a content folder,

1. In the **Manage | Content** tree, navigate to the content that you want to edit.
2. Right-click the content and choose **Copy**.
3. Click **OK** in the Copy dialog.
4. Navigate to the repository location (folder or content) where you want to copy the content.
5. Right-click on the repository location (folder or content) where you want to copy the content item and select **Paste**.
6. 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 which application on your hard drive with which to preview the file.

To preview content:

1. In the main menu, select **Content > Content Management**.
2. View the **Manage | Content** tree by clicking **Content** at the top of the resource tree.
3. In the **Manage | Content** tree, navigate to the content that you want to edit. If your repository uses library services, your content may be available in the Workspace View.
4. In the **Summary** tab, click the **Preview** icon.
5. 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.

This section includes the following topics:

Re-Ordering Content When Browsing Content

You can reorder content in the Browse tab by modifying the Order column if the folder that you are browsing is configured to allow re-ordering of content, see "Re-Ordering Content" on page 9-9.

Searching for Content By Name

If you know the name of the content you need to find, you can search for it by name.

To search for content by name:

1. In the Manage | Content tree, select the repository you want to search.
2. In the right pane, select the **Search** tab.
3. In the Name field, select an operator from the drop-down list (contains, doesn't contain, and so on) and type the name that you wish to use.
4. Click **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, do the following:

1. In the Manage | Content tree, select the repository you want to search.
2. In the right pane, select the **Search** tab.
3. Click **Advanced Search Options** in the Search section.
4. Select whether you want to search **Published** content or **New (unpublished)** content by marking the appropriate check box. Mark both check boxes if you want to search for all repository content.
5. Click **Or Search Specific Properties**.
6. Specific the properties for which you want to search, in the **Search Specific Properties** section. Use Table 9-3 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 9-3 Specific Properties Search Criteria

Search Criteria	Usage
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 drop-down list.
Include Subtypes	If you want to search for individual property values within the content type(s) you selected, mark **Include Subtypes**.
Type Properties	Define the property values for which you want to search. Select a property from the drop-down list, choose an operator (contains, equals, and so on) to use, and enter the value you are searching for.

Table 9-3 Specific Properties Search Criteria

Search Criteria	Usage
Dates	You can search according to the date content was created or modified. Select **Created** or **Modified** from the drop-down list, select an operator (before, after, and so on), and use the calendar tool to select a date for which to search.
Creator or Editor	You can search for content according to the user name that created it or modified it. Select **Created By** or **Modified By** from the drop-down list and enter a user name in the provided field.
Binary Content	You can search for the binary file associated with content. For example, you can search for the size or the name of the file you want to find. Using the drop-down list, select the property of the binary file you want to search (file name, file size, or MIME type), select an operator and provide a value.
Path to Content	You can search for content according to the folder or sub folder that it resides in within your repository. This is referred to as the Path to Content. Fill in the folder name(s) you want to search using the following format: `/foldername/subfoldername/contentname`

7. When finished entering criteria, click **Search**. Your results display in the Search Result**s** section.

Searching the Full Text of Content

You can search the full text of a content item which includes both its associated property values as well as the associated content file. Full-text search can only be used to find repository content that has been published for delivery to your portal.

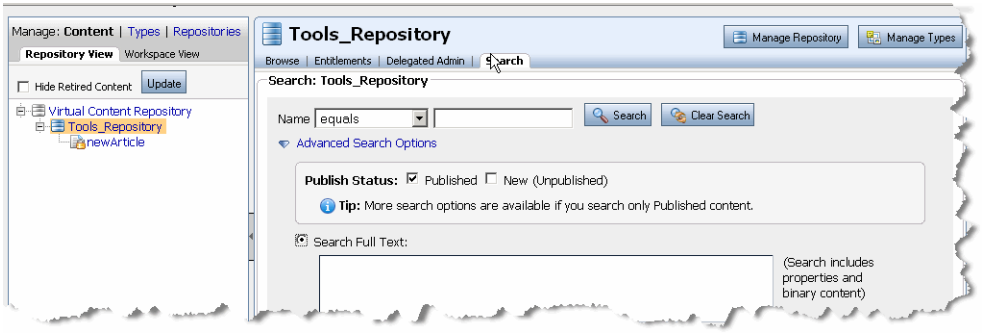
For example, you can use full-text search to find content of different content types that might include the same information.

To use a full-text search to find content, do the following:

1. In the Manage | Content tree, select the repository you want to search.
2. In the right pane, select the **Search** tab.

3. Click **Advanced Search Options** in the **Search** area. Figure 9-5 provides an example of the **Search** area.

Figure 9-5 Search Tab within the Repository View



4. Mark **Published** to search for content that has been published for delivery to your portal. Full-text search does not work on unpublished content.
5. Mark **Full Text Search**.
6. In the text box provided, type the keywords for which you want to search. The text you provide will be used to search both the property values for content and the associated binary files.
7. Click **Search**.

Using Expressions to Search for Content

You can use expressions to search for content within your BEA repository, see *WebLogic Portal Interaction Guide* for a complete reference to building content queries.

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 Content Workspace tab.

This section includes the following topics:

- Checking Out Content

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, select **Content > Content Management**.
2. Select **Manage | Content** to view the content resource tree.
3. From the Repository View, navigate to the content you want to check out.
4. In the Versioning & Workflow section of the Summary tab, click **Check Out**.
5. To edit your content, navigate to the Workspace View where the content item will be listed in the Checked-Out Items folder.

To check out content from the Workspace View:

1. From the main menu, select **Content > Content Management**.
2. Select Manage | Content to view the content resource tree.
3. In the Workspace View, navigate to the content you want to check out. (**My Workspace > Assigned Items >***content_name*).
4. In the Versioning & Workflow section of the Summary tab, click **Check Out**.
5. To edit your content, navigate to the content item within the Checked Out Items folder.

Your content item is now displayed in the **Workspace View** under the **Checked Out Items** folder. Use the steps listed in "Modifying Content" on page 9-7 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, select **Content > Content Management**.
2. Select Manage | Content to view the content resource tree.
3. In the Workspace View, navigate to the content you want to check in. (**My Workspace > Checked-Out Items >***content_name*).
4. In the Versioning & Workflow section of the Summary tab, click **Check In**.

Retiring Content

Retiring content means that it can no longer be used in search queries or displayed in your portal. When your BEA repository uses library services, 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 9-11.

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. From the main menu, select **Content > Content Management**.
2. In the Manage | Content resource tree, navigate to the content you want to retire.
3. In the Versioning & Workflow section of the Summary tab, click **Check Out**.
4. In the Versioning & Workflow section of the Summary tab, click **Check In** to check in the content with the Retired status.

5. In the Check In Content dialog, select the **Retired** status from the Check In with Status drop-down list.
6. Click **Check In**.

Viewing 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.

To view the version history of a content item:

1. From the main menu, select **Content > Content Management**.
2. In the Manage | Content tree, select the content item whose version history you want to search. You can use the Search tab, see "Searching for Content By Name" on page 9-15.
3. Select the **Version History** tab.

Searching Content Versions

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 search within the version history of a content item,

1. In the **Manage | Content** tree, select the content item whose version history you want to search. You can use the Search tab, see "Searching for Content By Name" on page 9-15.
2. Select the **Version History** tab.
3. Within the **Search** section, you can search the version comments field for the content item for specific text. Enter the text you want to find and click **Search**.
4. Optionally, you can use the **Advanced Search Options** to search the version history by workflow status, date modified or by the user name who modified the version you are looking for, see Table 9-4.

Table 9-4 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 the user name you specify.
Workflow Status	Searches for content versions that have the workflow status you specify.
Checked in	Searches for content versions that were checked in before or after the date you specify. You can also specify a single date.

5. After defining your search criteria, click **Search**.

Publishing a Different Version of Content

In some cases you may want to change the content version that is published within your portal. You can do this by checking out a previous version of a content item and transitioning 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.

1. Check out your content, see "Checking Out Content" on page 9-19.
2. Navigate to your content in the Checked-Out Items folder within the Workspace View tab.
3. Optionally, make any changes to the content such as updating the associated file and so on.
4. Click **Check In** to view the Check In dialog.
5. In the Check In dialog box, select the new published status from the drop-down list.
6. Click **Check In**.

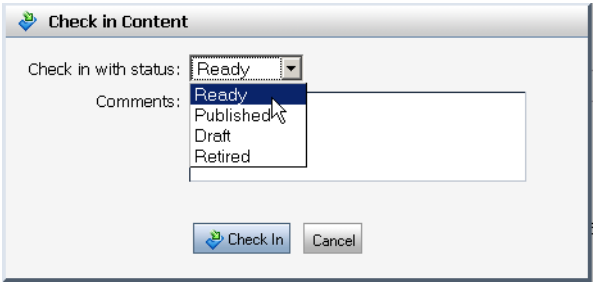
Using Content Workflows

If your repository is library services-enabled, you assign a status to your content each time a new version is created.

Content created in a library services-enabled BEA repository uses the default content workflow, unless a customized workflow has been implemented. WebLogic Portal's default content workflow includes the following statuses: Draft, Ready for Review, Rejected, Published and Retired, see Figure 9-6.

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

Figure 9-6 Changing Workflow Status when Checking In Content in the Portal Administration Console



This section assumes your repository uses the default content workflow and includes the following topic:

- Changing the Workflow of Content

Changing the Workflow of Content

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.

1. In the main menu, select **Content > Content Management**.
2. View the Manage | Content tree by clicking **Content** at the top of the resource tree.
3. In the Manage | Content tree, navigate to the content that you want to edit. Use either the Workspace View > My Assigned Items folder or the Repository View.

4. In the resource tree, click the content you want to edit to view the **Summary** page
5. In the Versioning and Workflow section of the Summary page, click **Check Out.**
6. After checking out the content, click **Versioning and Workflow** to view the Update Workflow dialog.
7. From the Update Workflow dialog, select a content workflow to use and click **Update**.
8. If finished modifying your content, click **Check In**.

CHAPTER 10

Delivering Content Within Your Portal

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

Retrieving and displaying content is typically done within the context of a JSP page. Within a JSP page, you can use the API directly, a JSP tag, or use content selectors or placeholders to retrieve content based on queries or rules.

You can use content selectors and campaigns deliver personalized content to user based upon personalization rules or conditions. Within this context, you use JSP tags to retrieve the content you need.

For more information about delivering personalized content, see the *WebLogic Portal Interaction Management Guide*.

This chapter includes the following sections:

Working with JSP Tags

WebLogic Portal provides JSP tags that can be used to 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.

This section includes the following topics:

- Retrieving Content with JSP Tags
- Displaying Content with JSP Tags

Retrieving Content with JSP Tags

There are four content-specific JSP tags that can be used to retrieve content. For more information about JSP tags, see the JSP Javadoc for Content Managment. Table 10-1 lists the content-related JSP tags.

Table 10-1 Content JSP Tags

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

Regardless of which tag you decide to use, retrieving for content within a JSP tag is done using one of two methods:

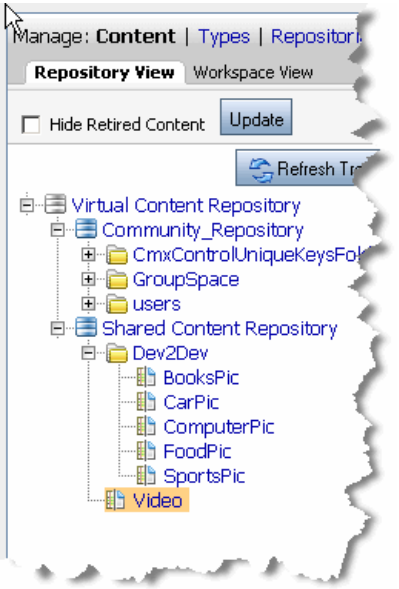
- Retrieving Content Using a Repository Path
- Using Queries to Retrieve Content

Retrieving Content Using a Repository Path

All content is addressable by a unique path. This path is visible in the 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

Portal Administration Console as shown in Figure 10-1, you could use the code in Listing 10-1 to retrieve `CarPic`.

Figure 10-1 Content Hierarchy



Listing 10-1 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 you will display it within the portal. Displaying content can be done using JSP tags and with personalization tools such as content selectors and placeholders. This section discusses JSP tags and provides some examples. For more information about personalizing content, see the *Interaction Management Guide*.

This section provides an overview of JSP tags you can use to display content.

Content Display JSP Tags

Content can be displayed in your portal using the tags listed in Table 10-2, "JSP Tags Used to Display Content," on page 10-5. For a complete reference for the tags listed, see the JSP Javadoc for Content Management.

Table 10-2 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 Workshop for WebLogic. 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 Workshop for WebLogic.

Table 10-2 JSP Tags Used to Display Content

JSP Tag	Description
`<dt:displaytemplate>`	Requests the registered view for a resource, which in turn renders it in the JSP. In order to use this tag, you must create display templates (views) and a mapping file (XML) to use templates with respective content types. For more information about setting up templates, see "Displaying Content with Display Templates" on page 10-6.
`<dt: displaycmtemplate>`	Requests the registered view for a content management resource, which in turn renders it in the JSP. In order to use this tag, you must create display templates (views) and a mapping file (XML) to use templates with respective content types. For more information about setting up templates, see "Displaying Content with Display Templates" on page 10-6.

Displaying Content with Display Templates

Display templates allow you to create views for content according to content type. You can create a default view based on a content type or you can create multiple views for that content type. For example, you can create a "thumbnail" view for an image as well as a view that displays the full-size image and a list of properties such as the price (if it is merchandise) or an address (if the image is an employee directory). You can set up display templates for any content type within your repository.

Views are based on JSPs that you create. Views are then defined in a configuration file that you create. After creating views and mapping them to a configuration file (XML), you can refer to those views (display templates) with a display template tag.

WebLogic Portal provides two types of display template tags. The `<dt:displaycmtemplate>` is provided to use to display resources from your content management system. The `<dt:displaytemplate>` tag should be used to display any other portal resources.

These tags are part of the `wlp-services-web-lib.war` which can be found in the *`//WebLogic_Home`*`/portal/lib/modules` directory.

This section includes the following topics:

Creating Views to Use with your Display Template

Views or templates are simply JSP pages that can be used to display associated resources. These pages can incorporate any JSP functionality you require.

As a best practice, it can be helpful to store all display templates in the same directory within your web application.

Creating a wlp-template-config.xml File

Before you can use a display template, you must create a configuration file that associates the views (JSP pages) you have created with the respective resources. For example, if you have created a view to use with a content type, you must configure that relationship using a `wlp-template-config.xml` file.

Listing 10-2 provides an commented outline of the `wlp-template-config.xml` schema.

Listing 10-2 wlp-template-config.xml

```
<wlp-display-template>
<!-- Describes the content repository's types and templates -->
<content-repository>
       <!-- Allows you to use a namespace to define multiple repositories if
needed -->
       <content-name-space>
       <!-- Name of repository. (Required) -->
       <name> </name>
       <!-- 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>
      <! -- Optionally, define a namespace to reference a content-resource --!>
      <namespace></namespace>
      <!-- Describes the content type (Optional) -->
      <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 (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>
            <!-- If the view name matches this template is used to render
content (Required) -->
            <uri> </uri>
            </view>
      </content-resource>
</content-name-space>
</content-repository>
<!-- Describes templates that are not linked to a content repository in any way
-->
```

```
<template-group>
<!-- Allows you to use namespaces if you want to define more than one template
group --!>
<template-name-space>
<name> </name>
<description> </description>
<default-template-uri> </default-template-uri>
<template>
       <description> </description>
<!-- Name of template. (Required) -->
       <name> </name>
       <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-display-template>
```

Table 10-3 provides an example of a valid `wlp-template-config.xml` file.

Listing 10-3 Sample wlp-template-config.xml Document

```
<wlp-display-template>
<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-display-template>
```

Using the <dt:displaycmtemplate> Within a JSP

The `<dt:displaycmtemplate>` is used to display resources from your content management system. 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 for an example of how the tag takes advantage of type inheritance.

This example includes three parts. Listing 10-4 shows an example JSP usage that uses a defined view while Listing 10-5 shows an example usage that uses the default view. Listing 10-6 provides the corresponding configuration file.

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 10-6 for an example. You can also use a * as <name> for the content-repository element.

Listing 10-4 <dt:displaycmtemplate> Tag Usage

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

Listing 10-5 <dt:displaycmtemplate> Tag Usage

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

Listing 10-6 Corresponding wlp-template-config.xml

```
<wlp-display-template>
<content-repository>
<name>MyRepo</name>
<default-template-uri>/MyDefault.jsp </default-template-uri>
<content-resource>
       <name>*</name>
<!-- Using an * in the name element defines a default view for content resources
--!>
       <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-display-template>
<!-- With this configuration the uri of "product/camera/smallview.jsp" will be
found. This is because a Digital Camera is a Camera and the camera 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.

This example includes two parts. Listing 10-7 shows the syntax of the JSP tag and Listing 10-8 provides the corresponding configuration file.

Listing 10-7 <dt:displaycmtemplate> Tag Usage

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

Listing 10-8 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-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-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

The `<dt:displaytemplate>` tag should be used to display any non-content management portal resources. See the *Commerce and Interaction Management JSP Tag Javadoc* for more information about this tag and its attributes.

Displaying Content with the Content Presenter Portlet

WebLogic Portal provides a content presenter portlet that can be used to preview content. For more information about the content presenter portlet, see " Using the Content Presenter Portlet" in the *Portlet Development Guide*.

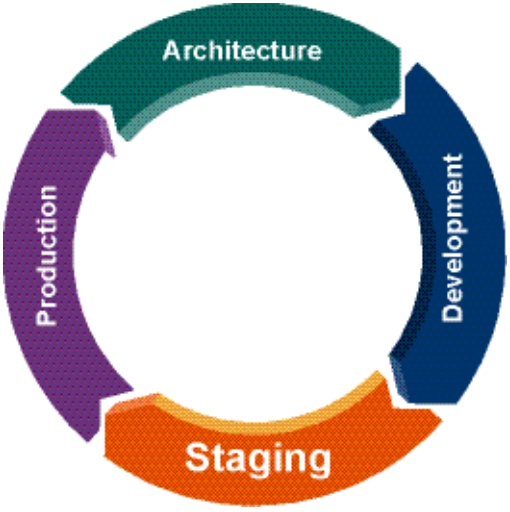
Part III Staging

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.

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 Figure 1-1:

Figure 1-1 Portal Life Cycle



Part IV Production

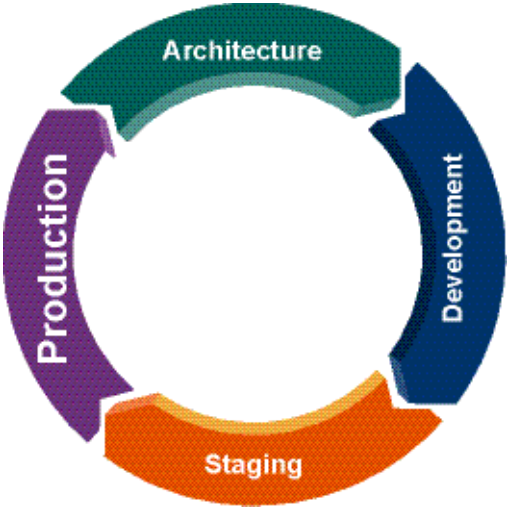
This section contains guidelines and procedures to manage your content after you have deployed your portal and are running in production.

Part IV contains instructions for tasks you can accomplish in the production phase and includes the following chapter:

- Chapter 12, “Caching Content”
- Chapter 13, “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 Figure 1-1:

Figure 1-1 Portal Life Cycle



CHAPTER 11

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 of a user with 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 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 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 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 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.

CHAPTER 12

Caching Content

After you have moved your portal application into production, you can use the Portal Administration Console to edit and maintain content within the content repository; you can update personalization features such as content selectors and placeholders; and you can 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 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**.
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.

CHAPTER 13

Managing Content Security

WebLogic Portal provides two ways to secure your content. 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

You can control how users and groups interact with the repository by associating delegated administration roles that are used to determine whether to grant or deny access to resources, and to determine which capabilities on those resources are available to the administrator.

Before starting to use the Virtual Content Repository, you can want to set up the users, roles, and privileges you want to use in order to dictate who can add, edit, or delete content within the repository.

Before you can assign 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 the repository or folder level, all children of that level inherit that delegated administration right.

However, 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*.

APPENDIX A

Importing Third-Party Content

The BulkLoader is a command-line application that loads content and metadata from a filesystem into a BEA Virtual Content Repository.

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 each specific content item, 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, then both the metadata and binary files are transferred to the repository. If you load into a filesystem repository, then only the metadata is transferred to the database--the actual content files remain in place on the filesystem.

Note: You cannot use the Bulkloader to update existing content (or its metadata) within a filesystem repository. It can only be used to add new content to the repository. As with other types of BEA repositories, you can add new and modify existing content using the Portal Administration Console.

This document 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, and prepare metadata files.

This section includes the following topics:

- Creating a Repository
- Creating Appropriate Types
- Preparing a Content Directory
- Preparing 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 ones provided. 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 both of these associations. In addition, this section describes how to add metadata when Library Services are enabled for your repository and how to name and store metadata files properly.

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 will be 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 filesystem 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:

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 tries 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 properties in an `dir.md.properties` file are inherited by content in subdirectories. You can override this behavior by specifying the `+recurse` flag (to turn off recursion) and the `+inheritProps` flag (to turn 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. This section includes the following sections:

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(s) is provided with Weblogic Portal:

- **Windows:** `\\WebLogic_HOME\wl_server\cm\bin\load_cm_data.cmd`
- **Unix:** `WebLogic_HOME/wl_server/cm/bin/load_cm_data.sh`

This section includes the following section:

- Editing the BulkLoader Script

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:** `\\WebLogic_HOME\wl_server\cm\bin\load_cm_data.cmd`
- **Unix:** `WebLogic_HOME/wl_server/cm/bin/load_cm_data.sh`

1. Open the script file for editing.
2. Set the `PLATFORM_HOME` variable to point to your WebLogic Server installation. For example:

   ```
   PLATFORM_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 in a directory called `images`, located in `D:\myContent\images`, then 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 with to load into the content repository. If you do not specify this option, then the current directory (.) is used. When bulkloading content to a filesystem repository, you must specify this option and the directory must match the `cm_fileSystem_path` property that was defined when the repository was created. The path specified with this option can be relative or absolute. For more information about filesystem repositories, see "Working with a BEA File System Repository" on page 3-7.
`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.

Table A-3 Required Bulkloader Parameters (Continued)

Required Parameters	Description
`-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 with to load into the content repository. If you do not specify this option, then the current directory (.) is used. This directory must match the `cm_fileSystem_path` property that was defined when the repository was created. The path specified with this option 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.
`-ignoreErrors`	Ignore errors while loading content (errors are still reported).
`+ignoreErrors`	If an error is encountered, stop precessing. [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.

Table A-4 Optional Bulkloader Parameters (Continued)

Optional Parameters	Description
`-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>`	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. This flag sets the class name of a LoaderFilter to run files through. This can be specified multiple times to add to the list of LoaderFilters. 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 that, it looks for a `loader.defFilters` property. This is the colon-separated list of LoaderFilter class names BulkLoader should always load. Unless that file is modified, BulkLoader will load an ImageLoaderFilter, which will pull the width and height from `*.gif`, `*.jpg`, `*.png`, and `*.xbm` image files.
`-filters`	Clears the current list of LoaderFilters, including the default filters.
`--`	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 as you wish, to suit your specific environment and needs.

Listing A-4 load_cm_data.cmd BulkLoader Example 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
```