BEA WebLogic Portal®

Repository Guide

Version 8.1 with Service Pack 4
WebLogic Portal 8.1 Repository Guide
Revised: December 2004

Contents

About This Document

This document discusses how to work with BEA repositories with WebLogic Portal.

Product Documentation on the dev2dev Web Site

BEA product documentation, along with other information about BEA software, is available from the BEA dev2dev Web site:

`http://dev2dev.bea.com`

To view the documentation for a particular product, select that product from the list on the dev2dev page; the home page for the specified product is displayed. From the menu on the left side of the screen, select Documentation for the appropriate release. The home page displays the complete documentation set for the product and release that you select.

Related Information

Remember that WebLogic Portal uses many components from the WebLogic Platform. See the following documentation for more information about tuning WebLogic Portal:

- WebLogic Portal 8.1 Content Management Guide
- WebLogic Portal 8.1 Bulkloader Guide
- Whitepaper: Integrating Content into the BEA Virtual Content Repository

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for BEA ProductName ProductVersion.

If you have any questions about this version of BEA ProductName, or if you have problems installing and running BEA ProductName, contact BEA Customer Support at **http://support.bea.com**. You can also contact Customer Support by using the contact information provided on the quick reference sheet titled "BEA Customer Support," which is included in the product package.

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

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

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.
`monospace text`	Indicates *user input*, as shown in the following examples: • Filenames: `config.xml` • Pathnames: *`BEA_HOME`*`/config/examples` • Commands: `java -Dbea.home=BEA_HOME` • Code: `public TextMsg createTextMsg(`
	Indicates *computer output*, such as error messages, as shown in the following example: `Exception occurred during event dispatching:java.lang.ArrayIndexOutOfBoundsException: No such child: 0`
`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `WebLogic Portal 8.1 Repository Guide`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
[ ]	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `java utils.MulticastTest -n` *`name`* `[-p` *`portnumber`*`]`
|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed. *Example*: `java weblogic.deploy [list|deploy|update]`

Convention	Item
`. . .`	Indicates one of the following in a command line: • That an argument can be repeated several times in a command line • That the statement omits additional optional arguments • That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name] [-f "file1.cpp file2.cpp file3.cpp . . ."`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER 1

Working with BEA Repositories

WebLogic Portal uses the Virtual Content Repository to access content repositories. The Virtual Content Repository can contain multiple content repositories, including third-party repositories. Because the Virtual Content Repository maintains connections with all associated content repositories (according to parameters you set), it allows you to federate your content management tasks across all repositories such as creating content types, delegated administration roles and content search.

Many Portal subsystems interact with the Virtual Content Repository, including:

- Content Management JSP tags that execute queries to deliver dynamic content to end users
- Content Selectors
- Campaigns that deliver dynamic, personalized content to users, based upon personalization rules or conditions

Architectural Overview of BEA Repositories

Generally, each BEA Repository uses the portal database to store content and associated content metadata. However, depending on your needs, this configuration can change. This section discusses Virtual Content Repository terminology, basic architectural points, and some diagrams to illustrate different types of repositories.

When To Create Repositories

You must create connections to any content repositories (third-party or BEA) before packaging your application into an EAR file. This means creating only the root repositories, not the content nodes and content items. After you create repositories, they are registered in the application-config.xml deployment descriptor. When you create the application EAR, application-config.xml becomes read-only and cannot be modified within the EAR. That is, you cannot add or remove repositories in the WebLogic Administration Portal when the application is in an EAR file.

For information on creating repositories, see the "Add a New Repository Connection" in the WebLogic Administration Portal help system at http://e-docs.bea.com/wlp/docs81/adminportal/help/CM_CreateNewRepository.html.

Table 1-1 BEA Repository Terminology

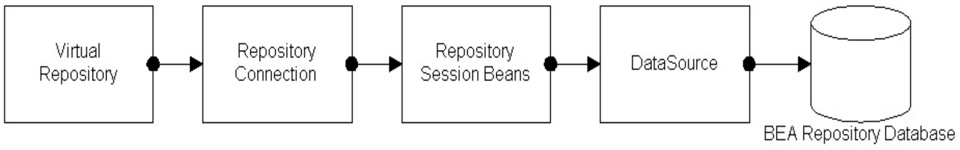
Term	Definition
Repository	The physical store for all folders (nodes), content items, properties and types. For a BEA Repository, this can be just a database, or a database plus a filesystem. A repository can also be a third-party or in-house content repository.
Repository Connection	The configuration information that describes how to connect and interact with a repository. This includes the SPI implementation information, the repository name and other additional properties specific to the repository implementation.
Database	The physical database used to store: • Content metadata • Content binaries (if you are not using a separate filesystem) • Published content (if using Library Services)
Filesystem	The physical location of the filesystem that contains the binary content files associated with the metadata stored in the database.
Virtual Content Database	The database tables used to store versioning and lifecycle information for non-published content, when using BEA's Library Services.

Architectural Summary

Within any repository configuration, the following guidelines apply:

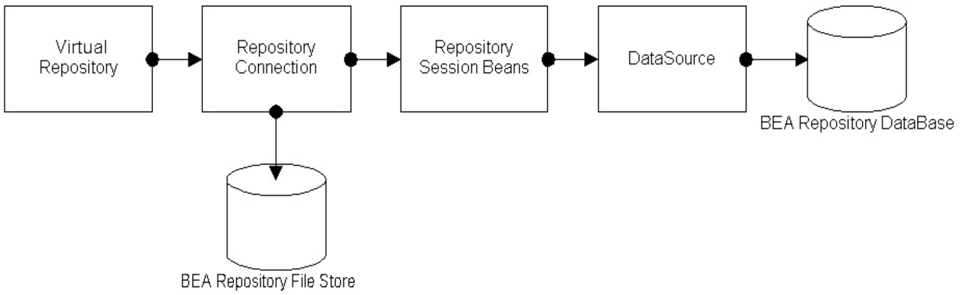
- Content metadata is always stored in the database.
- If you are using Library Services, additional database tables are used to store non-published binaries and their associated metadata.
- If you choose to use a filesystem to store your binary content (instead of a database), the metadata for those binaries will still be stored in the database.
- If you use a third-party repository and/or content management system, you will need to create a Repository Connection to that repository so it is "known" to the Virtual Content Repository.

Figure 1-1 Architecture of a Database-based BEA Repository



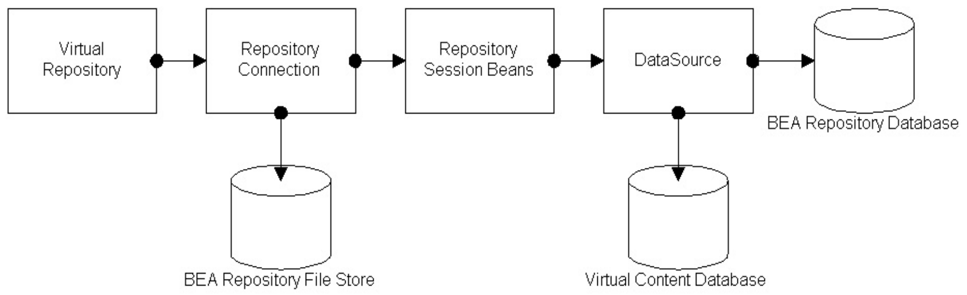
This is the simplest architecture used and outlines the structure of the default BEA Repository.

Figure 1-2 Architecture of a Filesystem-based BEA Repository



With a filesystem-based repository is the repository connection goes directly to the path of the repository filesystem where the binaries are kept.

Figure 1-3 Architecture of a Filesystem-based BEA Repository that uses Library Services



When a repository uses Library Services, separate database tables are used to store versioning and lifecycle information. Only the binaries with a status of "published" are stored in the filestore. For more information, see "Using Library Services with a Filesystem Repository" on page 1-5.

Working with a Filesystem Repository

Filesystem repositories allow you to store your binary content in a filesystem, yet still access that content with the functionality provided by the Virtual Content Repository.

Typically, this increases performance for both data retrieval within your portal and bulkloading large amounts of data (and its associated metadata) to a repository database within the WebLogic Portal Virtual Content Repository.

Using Content Types

There are two considerations for using content types within a fileystem repository.

- Content types must have ONE and only one binary property definition. This property definition must be declared as primary and required.
- In a filesystem repository, a hierarchy node (folder) cannot be associated with a content type. In other words, folders cannot be with associated with metadata.

Adding Content to a Filesystem Repository

When you create a filesystem repository, you use the Virtual Content Repository to add and delete content from the filesystem. The structure of the filesystem will mimic the structure of the repository. For example, creating a folder in a filesystem repository creates a folder in the shared directory.

Note: When organizing content within a filesystem repository, content items cannot contain other content items. Only content folders (hierarchy nodes) can contain content items.

Naming Content Items

When adding content, the name of the content item will match the name of the associated binary file, see "Modifying Content within a Filesystem Repository" on page 1-5. When you use the Bulkloader to add new content items, content items are named according to the binary file name associated with them.

Modifying Content within a Filesystem Repository

When you replace the binary file associated with a content item, the original binary file is deleted from the filesystem repository and replaced with the new file. The name of the content node is changed to match the name of the new file.

Warning: When you rename a content item, the associated binary file is also renamed. Because of this, be sure to always include the file extension when renaming content items within a filesystem repository.

Deleting Content from a Filesystem Repository

When you delete a folder or content item within a filesystem repository, both the binary file (stored in the filesystem) and the associated metadata (stored in the database) are deleted.

Using Library Services with a Filesystem Repository

A filesystem repository can be Library Services-enabled. When using Library Services, only content items with a "published" status are kept in the filesystem. Non-published versions of content items are stored in the database. This provides an additional safeguard if the filesystem gets damaged or removed. If you use Library Services with a filesystem repository, you automatically have a backup of all non-published binaries which are kept in the Virtual Content Database (see Figure 1-3, "Architecture of a Filesystem-based BEA Repository that uses Library Services," on page 1-4).

Bulkloader Considerations

When using Library Services with a filesystem repository, you cannot use the Bulkloader to update existing content (or its metadata).

When you add new content to a filesystem repository that is using library services, all new content will be created as checked out in Draft status under bulkloader's username.

Configuring a Filesystem 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 filesystem repository, the datastore is a filesystem on your network. When you add filesystem repository, you also need to add custom properties that direct WebLogic Portal to that filesystem.

The two steps required for configuring a filesystem repository are:

- Creating a connection to the new filesystem repository
- Using the Bulkloader to upload the metadata associated with the filesystem repository data

Before You Begin

The recommended way to implement a filesystem repository is to modify the properties of the default BEA repository. By default, the BEA repository class remains deployed even after you modify the connection information. This is intended because the BEA repository classes (com.bea.content.spi.internal.RepositoryImpl) are used to store metadata in the database.

Create a Connection to the New Filesystem Repository

1. Using the Portal Administration Portal, configure a connection to the soon-to-be new repository.
   a. View the **Manage Repositories** tree by selecting Repository from the View drop-down list.
   b. In the **Manage Repositories** tree, right-click the existing BEA repository.
   c. Select **Edit Repository** from the pop-up menu.
   d. In the **Editor** pane, provide the following information:

Input Field	What Is Needed:
Repository Name	Enter the name of the repository you are creating. For example, `FileRepository`.
Connection Class	If you are using a filesystem repository, the connection class is the following: `com.bea.content.spi.internal.FileSystemRepositoryImpl`
User Name	Enter the user name you want to use to login to this repository.
Password	Enter the password associated with the user name used to login to this repository.
Enable Library Services	Mark this checkbox if you want to use WebLogic Portal's library services for this repository. Library Services cannot used with third-party content management systems. If you want remove Library Services, you will need to create a new instance of the repository.

e. Click **Add Property**.

– In the **Key** field, enter **cm_fileSystem_path**.

– In the **Value** field, enter the path to the filesystem that contains your content. For example: `/home/myData`.

- Be sure that the your portal application has network access to your filesystem.

f. Optionally, if you are accessing your filesystem through a web server, you can add an additional property. Click **Add Property**.

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 via the `cm_fileSystem_webpath` property.

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

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

- Although not required, it is highly recommended that you set the `cm_fileSystem_webpath` property if your filesystem is exposed via a web server as it can significantly improve performance.

g. In the **Cache Properties** box, edit the cache properties, if necessary. You can also choose to accept the default cache settings.

h. Click **Create**.

Warning: Do not set the `STREAMING_ENABLED` property for a filesystem repository.

Use the Bulkloader to Upload Metadata for the Associated Filesystem

Note: This example uses the following sample names:

- The `cm_fileSystem_path` of the filesystem repository is `/home/myData`
- The name of the `repository` within the Virtual Content Repository is `FileRepository`.
- The name of the `filesystem directory` is `filesystem`.

1. Populate `/home/myData` with the directory that you want to upload.
2. Call the BulkLoader with the following arguments:

```
java com.bea.content.loader.bulk.BulkLoader -fileSystem -application <myApp>
-repository FileRepository /home/mydata/filesystem
```

   This will upload all the data into the filesystem repository.

3. Open the Virtual Content Repository within the Portal Administration portal to view your uploaded data. In this example, you would see a folder called **filesystem** below the **FileRepository** that would contain all the uploaded data.

Warning: If the Bulkloader fails during upload for any reason, you will need to delete the `cm_fileSystem_linked property` for the filesystem repository you created in the Virtual Content Repository. This property is temporary and used by the Bulkloader during upload. If the upload fails, delete this property before you try again.

For more detailed information about using the Bulkloader, see the Bulkloader Guide.

Working with a Third-Party Repository

For more information about working with third-party repositories, see http://e-docs.bea.com/wlp/docs81/whitepapers/vcr/index.html.

Creating Additional BEA Repositories

Your content management system can be made up of multiple BEA Systems repositories and/or multiple third-party repositories. You can create multiple content repositories to meet your unique business needs. For example, if you need a physical separation of your content data then you can create multiple BEA Repositories.

Note: You must be working within an XA domain to effectively use multiple repositories, if using Library Services.

Note: For large projects with several thousand content items, you may need to use separate database instances and minimize changes done in the production environment.

This process has five major steps:

- Step 1: Create Database Objects for the New Repository
- Step 2: Connect the New Repository to the Server
- Step 3: Associate your Portal Application with the New Repository
- Step 4: Start the Server
- Step 5: Create the New BEA Repository

Step 1: Create Database Objects for the New Repository

In this step you will create database objects for your additional content management database.

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 the following sample scripts which can be copied and used to define the database resources that must be configured proir to running any `.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 on creating databases.

- For **Oracle**, BEA provides the following sample scripts: `WL_HOME/portal/db/oracle/817/admin/create_tablespaces.sql` and `create_users.sql` .
- For **SQL Server** and **Sybase**, BEA provides the following sample scripts:
  - For SQL Server: `WL_HOME/portal/db/sql_server/2000/admin/create_database.sql`

    Note: The `WEBLOGIC_INDEX` file group must be defined for indexes created via the SQL Server `cm_create_tables.sql` and `cm_create_indexes.sql` scripts to execute without errors.
  - For Sybase: `WL_HOME/portal/db/sybase/125/admin/create_devices.sql` and `create_database.sql`

    Note: The `WEBLOGIC_INDEX` file group must be defined for indexes created via the Sybase `cm_create_tables.sql` and `cm_create_indexes.sql` scripts to execute without errors.
- For **DB2**, BEA provides the following sample scripts: `WL_HOME/portal/db/db2/8/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. Navigate to the appropriate database directory based on your environment. Depending on your database vendor, the path is:

   Oracle: `WL_HOME/portal/db/oracle/817`

Note: This path is used for both Oracle 8.1.7 and 9i.

SQL Server: `WL_HOME/portal/db/ql_server/2000`

Sybase: `WL_HOME/portal/db/sybase/125`

DB2: `WL_HOME/portal/db/db2/8`

4. Run the following scripts to create content management database objects:
   - `cm_create_tables.sql`
   - `cm_create_fkeys.sql`
   - `cm_create_indexes.sql`
   - `cm_create_triggers.sql`
5. Run the following scripts from the directory `WL_HOME/portal/db/data/required`:
   - `sample_cm_insert_system_data.sql`

This task is complete.

Step 2: Connect the New Repository to the Server

1. Start WebLogic Server for your domain, and login to the console.
2. Configure a new connection pool for your additional content management database.
   a. Go to Services -> JDBC -> Connection Pools.
   b. Right-click an existing XA Connection Pool and select Clone.

Note: For more information regarding XA connections, see

http://edocs.bea.com/platform/docs81/confgwiz/examples.html#1074297

   c. Choose a name for the new Connection Pool (For example: `contentPool2`).
   d. In the General tab, edit the settings for the cloned pool to match those of the new database objects you will be using for the additional repository.
   e. Click Clone, and apply the changes.
3. Update your data sources.
   a. From Services -> JDBC -> Data Sources, right-click a data source.
   b. Select "Clone..." from the drop-down menu.

c. Change the name of the data source from “Clone of ...” to something meaningful (For example: newContentDataSource).

d. Set the JNDI Name to match the new data source name.

e. Using the Pool Name drop down menu, set the pool to the new connection pool name (For example, contentPool2).

f. Click Clone, and apply the changes.

Step 3: Associate your Portal Application with the New Repository

The BEA Repository uses Entity Beans (EJBs) to associate a portal application with its data source(s). Because BEA’s default configuration is configured for just one data source (or repository), you’ll need to modify the XML files that maintain the EJB definitions for your data source(s). In short, you need to create new EJBs in these files that can be utilized by the additional repository you have created.

The easiest way to do this is to use the existing XML files as templates for creating the new EJB definitions and then add the new EJB definitions to the original file.

Redeploy Content `EJBs` to New Data Source (`ejb-jar.xml,weblogic-ejb-jar.xml,` and `the weblogic-cmp-rdbms-jar.xml`).

1. Create the workspace in which you will do your editing.

   a. Create two temporary, working directories to use when completing your edits. For example, `c:/temp/working` and `c:/temp/newfiles`

   b. Locate the `content_repo.jar` in the portal application directory that you want to associate with the new repository. For example, *`WL_HOME`* `/weblogic81/samples/portal/portalApp/content_repo.jar`

   c. Place a copy of the content_repo.jar file in each of the working directories you have created.

   d. Using a compression utility such as WinZip or JavaJar, unzip each content_repo.jar in its respective directory. At this point, the content_repo.jar files and their contents are identical.

   e. You will now work with three files:

   – `ejb-jar.xml`

– `weblogic-ejb-jar.xml`
– `weblogic-ejb-rdbms-jar.xml`

You will make edits to the files contained in your `C:/temp/working` directory and copy those changes to the files contained in your `C:/temp/newfiles` directory.

2. After creating the temp directories and files to work with, configure the `ejb-jar.xml` file.

   a. Open the `C:/temp/working/ejb-jar.xml` file and open the `C:/temp/newfiles/ejb-jar.xml` file. You will have two files open.

   b. In the `C:/temp/working/ejb-jar.xml` file, find and replace, `EJB` (`EJB` is case sensitive) with a new name. For example: `EJBNew`.

   Note: You will need to use this new name throughout all of the files.

   c. Copy the following from the `C:/temp/working/ejb-jar.xml` file to the `C:/temp/newfiles/ejb-jar.xml` file.

Copy everything between . . . from your `c:/temp/working/ejb-jar.xml` **file**	**and paste before . . . in your** `c:/temp/newfiles/ejb-jar.xml` **file**
`<enterprise-beans> </enterprise-beans>`	`</enterprise-beans>`
`<relationships> </relationships>`	`</relationships>`
`<assembly-descriptor> </assembly-descriptor>`	`</assembly-descriptor>`

   d. Save the `C:/temp/newfiles/ejb-jar.xml`.

3. Using the temporary directories and files you created in step 1. configure the `META-INF/weblogic-ejb-jar.xml` file.

   a. Open the `C:/temp/working/META-INF/weblogic-ejb-jar.xml` file and the `C:/temp/newfiles/ META-INF/weblogic-ejb-jar.xml` file. You will have two files open.

   b. In the `C:/temp/working/META-INF/weblogic-ejb-jar.xml` file , find and replace, EJB (EJB is case sensitive) with a new name. For example: EJBNew.

   Note: Use the same name you used in Step 2b.

c. In the `C:/temp/working/META-INF/weblogic-ejb-jar.xml` file , find and replace, (case sensitive) `BEA_content` with a new name (for example, `BEA_content_new`.)

d. In the `C:/temp/working/META-INF/weblogic-ejb-jar.xml` file, find and replace `<jndi-name>contentDataSource </jndi-name>` (case sensitive), and replace with DataSource you defined in Step 2: Connect the New Repository to the Server. (For example, `<jndi-name>newContentDataSource</jndi-name>`).

e. Copy the following from the `C:/temp/working/META-INF/weblogic-ejb-jar.xml` file to the `C:/temp/newfiles/META-INF/weblogic-ejb-jar.xml`:

Copy everything between . . . from your `C:/temp/working/META-INF/weblogic-ejb-jar.xml` **file**	**and paste before . . . in your** `C:/temp/newfiles/META-INF/weblogic-ejb-jar.xml` **file**
`<weblogic-ejb-jar> </weblogic-ejb-jar>`	`</weblogic-ejb-jar>`

f. Save the `C:/temp/newfiles/META-INF/weblogic-ejb-jar.xml` file.

4. Using the temporary directories and files you created in step 1., configure the `META-INF/weblogic-cmp-rdbms-jar.xml` file.

a. Open the `C:/temp/working/weblogic-cmp-rdbms-jar.xml` file and the `C:/temp/newfiles/META-INF/weblogic-cmp-rdbms-jar.xml` file. You will have two files open.

b. In the `C:/temp/working/weblogic-cmp-rdbms-jar.xml` file, find and replace `EJB` (`EJB` is case sensitive) with same name that you used in the //`META-INF/weblogic-ejb-jar.` For example: `EJBNew.`

Note: Use the same name you used in Step 2b.

c. In the `C:/temp/working/weblogic-cmp-rdbms-jar.xml` temp file, find and replace, `contentDataSource` (case sensitive) with the DataSource you defined in "Step 2: Connect the New Repository to the Server.". (For example: newContentDataSource)

d. Copy the following from `C:/temp/working/weblogic-cmp-rdbms-jar.xml` file to the `C:/temp/newfiles/META-INF/weblogic-cmp-rdbms-jar.xml` file:

Copy everything starting at the first . . . and ending at the last . . . from the `C:/temp/working/META-INF/weblogic-cmp-rdbms-jar.xml`**file**	**and paste after the last . . . in the** `C:/temp/newfiles/META-INF/weblogic-cmp-rdbms-jar.xml` **file**
`<weblogic-rdbms-bean>` `</weblogic-rdbms-bean>`	`</weblogic-rdbms-bean>`
`<weblogic-rdbms-relation>` `</weblogic-rdbms-relation>`	`</weblogic-rdbms-relation>`

5. In the `C:/temp/newfiles` directory, re-zip or re-compress the content_repo.jar file. Overwrite the existing content_repo.jar file with the new file of the same name that contains your edited XML files. Be sure to keep the files in their original directory structure, including the `META-INF` directory.
6. Copy the new content_repo.jar file you just created back into the original install directory. For example `WL_HOME/weblogic81/samples/portal/portalApp/`

Step 4: Start the Server

Restart the WebLogic Server to redeploy `content_repo.jar`.

Step 5: Create the New BEA Repository

To add a new repository:

1. Open the Administration Portal. (Start -> Programs -> BEA WebLogic Platform 8.1 -> Examples -> WebLogic Portal -> WebLogic Administration Portal)
2. Select “Content” in the menu at the top of the screen.
3. In the Browse and Edit Content Resource tree to the left, select Repository from the View drop-down menu.
4. In the Manage Repositories tree to the left, right click on the Virtual Content Repository.
5. Select Add Repository from the pop-up menu (or click the Add Repository button in editor pane).
6. In the Editor pane, provide the following information:

Field	**Description**
Repository Name	The name you give your new repository. For example: `MyNewRepository`
Connection Class	(`com.bea.content.spi.internal.RepositoryImpl`)

Note: When you create a new repository, if you cut and paste the properties from a text pad document, be sure to delete any trailing blanks.

7. Click Add Property, and provide the following parameters:

Key	**Value**
`NODE_OPS_HOME`	`${APPNAME}.BEA_content_`<the name you chose, in our example, it was `new>.RepoNodeOpsHome` For example: `${APPNAME}.BEA_content_RepoNodeOpsHome`
`OBJECT_CLASS_OPS_HOME`	`${APPNAME}.BEA_content_`<the name you chose, in our example, it was `new>.RepoObjectClassOpsHome`
`SEARCH_OPS_HOME`	`${APPNAME}.BEA_content_`<the name you chose, in our example, it was `new>.RepoSearchOpsHome`

8. Click Create.

APPENDIX A

Using 7.x Repositories with 8.x Portals

WebLogic Portal 7.0, BEA provided the capability to bulkload content from third-party repositories into the BEA content database. Various changes in WebLogic 8.1, including database schema changes and API changes, prevent automatically bulkloading 7.0 repositories into the new 8.1 repository architecture.

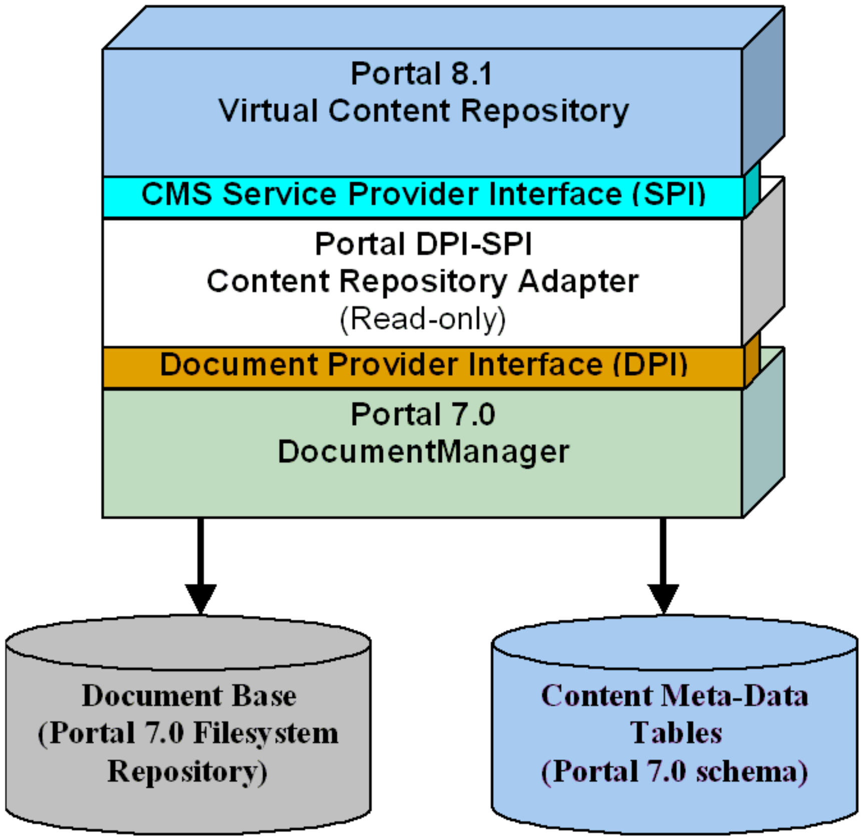
If you need to continue to use 7.0-compatible repositories, BEA provides a content repository adapter with 8.1 that allows you to bulkload 7.0 content to the newer 8.1 repository architecture.

It is important to note that you must use WebLogic 8.1 tag libraries to work with content within an 8.1 portal. The 7.0 tags are not supported with 8.1

In addition, if you are using a 7.0-compatible repository, 8.1 repository management features are not available and cannot view your 7.0 repository via the Portal Administration tools.

For an graphic overview of how integration works with the adapter, see Figure A-1.

Figure A-1 Architectural Overview of Integrating 7.0 Repositories with a 8.1 Virtual Content Repository



Configuring a 7.0 repository for use with an 8.1 Portal

Configuring a 7.0 repository to use the content adapter requires the following steps:

- Step 1: Edit Deployment Descriptor
- Step 2: Create 7.0 Content Tables within the 8.1 Database
- Step 3: Create Filesystem Document Directory
- Step 4: Loading Content with Bulkloader

Note: These tasks assume you are NOT working with an EAR file. If you are working in an EAR file, you will need to explode the EAR file in order to make the necessary configuration changes.

Step 1: Edit Deployment Descriptor

The <WL_HOME_8.1><portalAPP_name>/META-INF/application-config.xml file is the deployment descriptor that contains repository configuration. Locate this file within your installation and use an XML editor or text editor to do the following

- Register the DocumentManager
- Register the Content Repository Adapter

Register the DocumentManager

You need to register the DocumentManager with the respective 8.1 application. To do this, add the configuration information for the 7.0 CMS Repository immediately before the `<ContentManagement/>` element (within the `<WL_HOME_8.1><portalAPP_name>/META-INF/application-config.xml` file).

Note: This sample text assumes the default settings. Refer to your existing 7.0 configuration files for any information specific for your deployment. For example, you'll need to change this text if you have customized or renamed the default manager and connection pool names.

```
<!-- Portal 7.0 CMS Repository -->

       <DocumentManager

             Name="default"

             DocumentConnectionPoolName="default"

             PropertyCase="none"

             MetadataCaching="true"

             MetadataCacheName="documentMetadataCache"

             UserIdInCacheKey="false"

             ContentCaching="true"

             ContentCacheName="documentContentCache"

             MaxCachedContentSize="32768"
```

```
            >
            </DocumentManager>

            <DocumentConnectionPool
              Name="default"
              DriverName="com.bea.p13n.content.document.jdbc.Driver"

URL="jdbc:beasys:docmgmt:com.bea.p13n.content.document.ref.RefDocumentProv
ider"
Properties="jdbc.dataSource=weblogic.jdbc.jts.commercePool;schemaXML=.\dms
Base/doc-schemas;docBase=.\dmsBase"
              InitialCapacity="20"
              MaxCapacity="20"
              CapacityIncrement="0"
            />
<!-- /Portal 7.0 CMS Repository -->
```

Register the Content Repository Adapter

You need to register the Content Repository adapter for the respective portal application.

To do this, you'll need to add the content store name and class to the end of the existing `<ContentManagement/>` element within the `<WL_HOME_8.1><portalAPP_name>/META-INF/application-config.xml` file.

Here is an example content store name and class:

Note: This sample text assumes the default settings. Refer to your existing 7.0 configuration files for any information specific for your deployment. For example, you'll need to change this text if you have customized or renamed the default manager and connection pool names.

```
<ContentStore Name="adapter"
ClassName="com.bea.p13n.content.adapter.RepositoryImpl"
```

```
Properties="CONTENT_MANAGER_HOME=${APPNAME}.BEA_personalization.DocumentMa
nager"/>
```

Step 2: Create 7.0 Content Tables within the 8.1 Database

In this step, you will create the necessary database tables to support storing 7.0 content within your 8.1 database. It is recommended that a DBA perform this step, for more information about how to run database scripts, see the respective vendor database sections in the Database Administration Guide.

Specifically, you'll need to create the Portal 7.0 content schema tables (DOCUMENT and DOCUMENT_METADATA) for the DocumentManager(s) you are using.

To do this, create an SQL script containing the 7.0 content schema information and run that script against the 8.1 database. The SQL extract below shows the table and key creation, as well as the index creation:

```
CREATE TABLE DOCUMENT (
    ID VARCHAR2(254) NOT NULL,
    DOCUMENT_SIZE NUMBER(15) NOT NULL,
    VERSION NUMBER(15) NULL,
    AUTHOR VARCHAR2(50) NULL,
    CREATION_DATE DATE NULL,
    LOCKED_BY VARCHAR2(50) NULL,
    MODIFIED_DATE DATE NULL,
    MODIFIED_BY VARCHAR2(50) NULL,
    DESCRIPTION VARCHAR2(2000) NULL,
    COMMENTS VARCHAR2(2000) NULL,
    MIME_TYPE VARCHAR2(100) NOT NULL)
;
CREATE TABLE DOCUMENT_METADATA (
    ID VARCHAR2(254) NOT NULL,
    NAME VARCHAR2(240) NOT NULL,
    STATE VARCHAR2(50) NULL,
    VALUE VARCHAR2(2000) NULL)
;
ALTER TABLE DOCUMENT
   ADD CONSTRAINT PK_DOCUMENT
   PRIMARY KEY (
      ID
   )
;D
ALTER TABLE DOCUMENT_METADATA
```

```
   ADD CONSTRAINT PK_DOCUMENT_MD
   PRIMARY KEY (
      ID,
   NAME
   )
;
ALTER TABLE DOCUMENT_METADATA
   ADD CONSTRAINT FK1_DOCUMENT_MD
   FOREIGN KEY ( ID)
   REFERENCES DOCUMENT (ID)
   ON DELETE CASCADE
;
CREATE  INDEX IX1_DOCUMENT ON DOCUMENT
(
  DOCUMENT_SIZE  ASC
)
;
CREATE  INDEX IX2_DOCUMENT ON DOCUMENT
(
  MIME_TYPE  ASC
)
;
CREATE  INDEX IX3_DOCUMENT ON DOCUMENT
(
  MODIFIED_DATE  ASC
)
;
CREATE  INDEX IX1_DOCUMENT_MD ON DOCUMENT_METADATA
(
  NAME  ASC
)
;
CREATE  INDEX IX2_DOCUMENT_MD ON DOCUMENT_METADATA
(
  STATE  ASC
)
;
```

Step 3: Create Filesystem Document Directory

Copy or create the content directory (e.g. dmsBase) for each DocumentManager to the Portal 8.1 domain, as for Portal 7.0. You will use this directory when you bulkload 7.0 content to the 8.1 repository.

Step 4: Loading Content with Bulkloader

After you have configured the content adapter, you can now bulkload content from a 7.0 repository into the 8.1 Virtual Content Repository. For more information about using the Bulkloader, see the Bulkloader Guide.

1. Copy the loaddocs.properties file from a Portal 7.0 domain directory and update the database details and connection pool names as required. The following example is for the default DocumentManager using Oracle 9i:

```
weblogic.jdbc.connectionPool.commercePool=\
        url=jdbc:weblogic:rmi,\
        driver=weblogic.jdbc.rmi.Driver,\
        props=weblogic.user=weblogic;weblogic.credential=weblogic;\
              weblogic.server.url=t3://localhost:7001;\
              weblogic.jdbc.datasource=weblogic.jdbc.jts.commercePool,\
        verbose=true

weblogic.jdbc.connectionPool.directToDB=\
        url=jdbc:oracle:thin:@dbserver6:1521:DBS6OR9U,\
        driver=oracle.jdbc.OracleDriver,\
props=user=user1;password=user1;server=jdbc:oracle:thin:@dbserver1:1521:DB
S1OR9U;password=user1
```

Note: When loading data into the repository all metadata is stored as properties and types. In addition to the user-defined types and properties there are explicit (system) content properties. These are documented in the WebLogic Workshop help system on edocs (http://e-docs.bea.com/workshop/docs81/doc/en/core/index.html). Most of these properties will return information about data loaded via the content adapter, however some will not. Below is a table showing which properties can and cannot be used.

Table A-1 Content Adapter Supported System Properties

Property	API	Description	Use with Content adapter
`cm_uid`	`Node.id.uid`	The unique ID for a content item. You can view a content item's unique ID by selecting the content item in the WebLogic Administration Portal and viewing the description in the Edit Content window.	Yes
`cm_createdDate`	`Node.createDate`	The date on which a content item was created. You can view Creation Date information for content items by selecting their content folder in the WebLogic Administration Portal and selecting the Browse Children page.	Yes
`cm_createdBy`	`Node.createdBy`	The user who created the content item. You can view Created By information for content items by selecting their content folder in the WebLogic Administration Portal and selecting the Browse Children page.	Yes
`cm_modifiedDate`	`Node.ModifiedDate`	The date the content item was last modified. You can view Modified Date information for content items by selecting their content folder in the WebLogic Administration Portal and selecting the Browse Children page.	Yes
`cm_nodeName`	`Node.name`	The name of the content item, as shown the WebLogic Administration Portal.	Yes
`cm_path`	`Node.path`	The Virtual Content Repository path to the content item. For example, '/BEA Repository/juvenilebooks/TheCrazyA dventure'.	Yes

Table A-1 Content Adapter Supported System Properties

`cm_isHierarchy`	`Node.type ==Node.HIERARCHY`	Identifies a content folder rather than a content item. Content folders can contain content items and child content folders. When using this property in queries, compare it using a Boolean value of true or false.	No
`cm_isContent`	`Node.type == Node.CONTENT`	Identifies a content item rather than a content folder. Content items contain properties from the type with which they are associated. When using this property in queries, compare it using a Boolean value of true or false.	No
`cm_objectClass`	`Node.objectClass.name`	The content type associated with a content item. You can view Type information for content items by selecting their content folder in the WebLogic Administration Portal and selecting the Browse Children page, or by selecting a content item and viewing its header row on the Edit Content page.	No
`cm_contentType`	`BinaryValue.contentType`	The mime type for content item binary properties. For example, 'image/jpeg'. You can view the mime type of a content item by selecting it in the WebLogic Administration Portal and looking at the binary property information.	Yes

Table A-1 Content Adapter Supported System Properties

cm_binarySize	BinaryValue.size	(For Node Properties)The size of the binary value of content items. You can view the size of a content item's binary value by selecting the content item in the WebLogic Administration Portal, clicking Download File on the binary property, and right-clicking the displayed binary file to view the file properties. When using this property in a query, specify binary size in bytes.	Yes
cm_binaryName	BinaryValue.name	(For Node Properties)The file name of the binary value of a content item. You can view the name of a content item's binary value by selecting the content item in the WebLogic Administration Portal and viewing the File Name value.	Yes

2. Start the Server:
3. Run the Bulkloader.

   Run the Portal 7.0 Bulkloader against the document base (dmsBase) in the Portal 8.1 domain. The following script illustrates an example Bulkloader script (loaddocs.bat) for the default DocumentManager:

```
call .\setDomainEnv.cmd

set CLASSPATH=%CLASSPATH%; C:\wl8sp4\weblogic81\portal\lib\wps\ejb\wps.jar


java -classpath %CLASSPATH% com.bea.p13n.content.document.ref.loader.BulkLoader
-verbose -ignore "doc-schemas" -ignore "Ads" -properties loaddocs.properties
-conPool commercePool -schema .\dmsBase\doc-schemas\doc-schema.xml -d .\dmsBase


java -classpath %CLASSPATH% com.bea.p13n.content.document.ref.loader.BulkLoader
-verbose -ignore "doc-schemas" -properties loaddocs.properties -conPool
commercePool -schema .\dmsBase\doc-schemas\ad-schema.xml -schemaName Ads -d
.\dmsBase Ads
```