



# BEA WebLogic Portal<sup>®</sup>

## BulkLoader Guide

Version 9.2  
Revised: June 2006

# Copyright

Copyright © 1995-2006 BEA Systems, Inc. All Rights Reserved.

## Restricted Rights Legend

This software is protected by copyright, and may be protected by patent laws. No copying or other use of this software is permitted unless you have entered into a license agreement with BEA authorizing such use. This document is protected by copyright and may not be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form, in whole or in part, without prior consent, in writing, from BEA Systems, Inc.

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

## Trademarks and Service Marks

Copyright © 1995-2006 BEA Systems, Inc. All Rights Reserved. BEA, BEA JRocket, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop, Built on BEA, Jolt, JoltBeans, SteelThread, Top End, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA AquaLogic, BEA AquaLogic Data Services Platform, BEA AquaLogic Enterprise Security, BEA AquaLogic Interaction, BEA AquaLogic Interaction Analytics, BEA AquaLogic Interaction Collaboration, BEA AquaLogic Interaction Content Services, BEA AquaLogic Interaction Data Services, BEA AquaLogic Interaction Integration Services, BEA AquaLogic Interaction Process, BEA AquaLogic Interaction Publisher, BEA AquaLogic Interaction Studio, BEA AquaLogic Service Bus, BEA AquaLogic Service Registry, BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Kodo, BEA Liquid Data for WebLogic, BEA Manager, BEA MessageQ, BEA SALT, BEA Service Architecture Leveraging Tuxedo, BEA WebLogic Commerce Server, BEA WebLogic Communications Platform, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Java Adapter for Mainframe, BEA WebLogic JDriver, BEA WebLogic Log Central, BEA WebLogic Mobility Server, BEA WebLogic Network Gatekeeper, BEA WebLogic Personalization Server, BEA WebLogic Personal Messaging API, BEA WebLogic Platform, BEA WebLogic Portlets for Groupware Integration, BEA WebLogic Real Time, BEA WebLogic RFID Compliance Express, BEA WebLogic RFID Edge Server, BEA WebLogic RFID Enterprise Server, BEA WebLogic Server Process Edition, BEA WebLogic SIP Server, BEA WebLogic WorkGroup Edition, BEA Workshop for WebLogic Platform, BEA Workshop for JSP, BEA Workshop Struts, BEA Workshop Studio, Dev2Dev, Liquid Computing, and Think Liquid are trademarks of BEA Systems, Inc. Accelerated Knowledge Transfer, AKT, BEA Mission Critical Support, BEA Mission Critical Support Continuum, and BEA SOA Self Assessment are service marks of BEA Systems, Inc.

All other names and marks are property of their respective owners.

# Contents

## Using BulkLoader

Overview .....	1
Preparing to Use BulkLoader .....	2
Creating a Repository .....	2
Creating Appropriate Types .....	2
Preparing a Content Directory .....	3
Preparing Metadata Files .....	3
Defining Metadata for a Directory of Files .....	3
Defining Metadata for Specific Files .....	4
Metadata Guidelines .....	4
Creating Metadata for a Library Services Enabled Repository .....	5
Naming and Storing Metadata Files .....	6
Metadata Summary .....	6
Configuring and Running BulkLoader .....	6
BulkLoader Command Examples .....	8
BulkLoader Parameter Reference .....	9
Example BulkLoader Script .....	12



# Using BulkLoader

## Overview

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

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 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.

If you want to load or transfer content that uses advanced repository features available in WebLogic Portal 9.2, you can use the Propagation Tool.

**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 the [WebLogic Portal Content Management Guide](#).

## 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 [WebLogic Portal Content Management Guide](#).

**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 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 is provided with Weblogic Portal:

**Windows:** `Weblogic92\portal\bin\load_cm_data.cmd`

**Unix:** `Weblogic92/portal/bin/load_cm_data.sh`

This section includes the following sections:

- [Preparing Your Environment](#)
- [Editing the BulkLoader Script](#)

**Note:** WebLogic server must be running when you use BulkLoader.

## Preparing Your Environment

BulkLoader scripts require access to two JAR files: the `content.jar` file and the `content_system.jar`. By default, these files are located in the `wlp-services-app-lib.ear` file. If you want to use the BulkLoader, you need to extract the `content.jar` and the `content_system.jar` files from the `wlp-services-app-lib.ear` file. After extracting the file, you also need to add their location to your classpath. You can delete these JAR file after you are finished using the BulkLoader.

1. Navigate to the `wlp-services-app-lib.ear` file. For example,
 

```
c:\bea\weblogic92\portal\lib\modules\wlp-services-app-lib.ear.
```
2. Use JavaJar or WinZip to extract the `content.jar` and the `content_system.jar` files from the `wlp-services-app-lib.ear`. Extract the `.JAR` files to a temporary location.
3. Add the `content.jar` and the `content_system.jar` files to the classpath on the system where you intend to use the 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:** Weblogic92\portal\bin\load\_cm\_data.cmd

**Unix:** Weblogic92/portal/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\weblogic92
```

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 1-1](#) shows a sample configuration.

### Listing 1-1 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 2](#).

**Table 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.

Parameter	Description
-d	<p>Specifies the base directory that contains the directories and files you wish to load into the content repository. If you do not specify this option, then the current directory (.) is used.</p> <p>When bulkloading content to a filesystem repository, you must specify this option and the directory must match the <code>cm_fileSystem_path</code> property that was defined when the repository was created. The path specified with this option can be relative or absolute.</p> <p>For more information about repository properties, see <a href="#">Working with a BEA Filesystem Repository</a> in the <a href="#">Content Management Guide</a>.</p>
<code>file1...fileN</code>	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 1-2 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 1-3](#) 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 1-3 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 3 Required Bulkloader Parameters**

Required Parameters	Description
<code>-repository &lt;repository name&gt;</code>	The name of the repository to run the loader against.
<code>-application &lt;app name&gt;</code>	The name of the application to run the loader against.
<code>-url &lt;wls url&gt;</code>	The WebLogic Server instance host where the content manager is running.
<code>-user &lt;principal username&gt;</code>	The username for the principal permitted to access the Loader EJB resource.
<code>-password &lt;principal password&gt;</code>	The password for the principal permitted to access the LoaderEJB resource.
<code>-d &lt;dir&gt;</code>	Specifies the base directory that contains the directories and files you wish to load into the content repository. If you do not specify this option, then the current directory (.) is used. This directory must match the <code>cm_fileSystem_path</code> property that was defined when the repository was created. The path specified with this option can be relative or absolute.

**Table 4 Optional Bulkloader Parameters**

<b>Optional Parameters</b>	<b>Description</b>
<code>-recurse</code>	Recurse into directories. [Default]
<code>+recurse</code>	Do not recurse into directories.
<code>-metaparse</code>	Parse HTML files for META tags. [Default]
<code>+metaparse</code>	Do not parse HTML files for META tags.
<code>-hidden</code>	Ignore hidden files and directories. [Default]
<code>+hidden</code>	Include hidden files and directories.
<code>-inheritProps</code>	Inherit metadata properties when recursing. [Default]
<code>+inheritProps</code>	Do not inherit metadata properties when recursing.
<code>-ignoreErrors</code>	Ignore errors while loading content (errors are still reported).
<code>+ignoreErrors</code>	If an error is encountered, stop preprocessing. [Default]
<code>-htmlPat &lt;pattern&gt;</code>	Specifies file extensions that represent HTML files. If the <code>-metaparse</code> flag is set, the values of <code>&lt;meta&gt;</code> and <code>&lt;title&gt;</code> 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, <code>*.htm</code> and <code>*.html</code> are used.
<code>-encoding &lt;encoding&gt;</code>	Specifies the file encoding to use. See your JDK documentation for the valid encoding names. [Default: the system's default file encoding]
<code>-match &lt;pattern&gt;</code>	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.
<code>-ignore &lt;pattern&gt;</code>	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.
<code>-mdext &lt;ext&gt;</code>	Specifies the file name extension for metadata property files. The value should start with a ".". This defaults to <code>.md.properties</code> .

**Table 4 Optional Bulkloader Parameters (Continued)**

Optional Parameters	Description
-filter <filter class>	<p>Although not commonly used, you can write a custom filter to set metadata values based on specific characteristics of a type of content. For instance, a filter might compute the width and height of an image file and set the values in metadata.</p> <p>This flag sets the class name of a LoaderFilter to run files through. 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 <code>content\com\bea\content\loader\bulk</code> file in the classpath. From that, it looks for a <code>loader.defFilters</code> property. This is the colon-separated list of LoaderFilter class names BulkLoader should always load. Unless that file is modified, BulkLoader will load an <code>ImageLoaderFilter</code>, which will pull the width and height from <code>*.gif</code>, <code>*.jpg</code>, <code>*.png</code>, and <code>*.xbm</code> image files.</p>
-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 <code>-Xms1000m</code> . Try using this flag if BulkLoader fails with an out of memory error.
-h	Display command line usage.

## Example BulkLoader Script

[Listing 1-4](#) 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 1-4 BulkLoader Script (Windows)

```
@ECHO OFF
REM
#####
REM #      (c) BEA SYSTEMS INC. All rights reserved
REM #
```



```

REM
#####

SETLOCAL

SET PLATFORM_HOME=C:\bea\weblogic92
FOR %%i IN ("%PLATFORM_HOME%") DO SET PLATFORM_HOME=%%~fsi
SET PORTAL_HOME=%PLATFORM_HOME%\portal
SET P13N_HOME=%PLATFORM_HOME%\p13n

CALL %PLATFORM_HOME%\common\bin\commEnv.cmd

@rem
*****
@rem Set any additional CLASSPATH information below
@rem
*****
setCLASSPATH=%POINTBASE_CLASSPATH%;%WEBLOGIC_CLASSPATH%;%P13N_HOME%\lib\
p13n_system.jar;%PORTAL_HOME%\lib\content.jar;%PORTAL_HOME%\lib\
content_system.jar;%CLASSPATH%

REM Set some defaults
if "%CM_DATA%"==" " set CM_DATA=..\db\data\sample\cm_data

%JAVA_HOME%\bin\java -classpath %CLASSPATH%
com.bea.content.loader.bulk.BulkLoader -verbose -repository "BEA
Repository" -application portalApp -d %CM_DATA% Ads

ENDLOCAL

```

---

## Using BulkLoader