BEA WebLogic Portal®

BulkLoader Guide

Version 8.1 with Service Pack 6
Revised: July 2006

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

Using BulkLoader

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 7
BulkLoader Command Examples 8
BulkLoader Parameter Reference 9
Example BulkLoader Script 12

Using BulkLoader

BulkLoader is a command-line application that loads content and metadata from a filesystem into a BEA Virtual Content Repository. This document explains how to use BulkLoader and includes these topics:

- Preparing to Use BulkLoader
- Configuring and Running BulkLoader
- BulkLoader Parameter Reference

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.

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, only the metadata is transferred to the database while the actual content files remain in place on the filesystem.

You can load any type of content using BulkLoader. The BulkLoader program is the only means by which you can batch load files into a repository.

Note: When you use the BulkLoader to delete content from a library services-enabled repository, the Bulkloader also deletes all version data associated with the deleted content.

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 discusses these preliminary steps.

Creating a Repository

BulkLoader loads content and metadata into a pre-established content repository. For information on creating a repository, see Creating a New Repository Connection.

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:

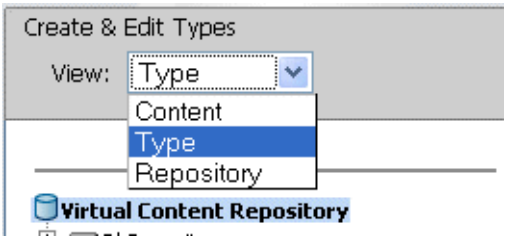
- Description
- File
- Image Name

You can create your own types or use those provided. For information on creating types, see Creating a New Content Type.

Note: When using a filesystem repository, at type associated with a content item must be created with *binary* as its primary property.

Tip: To view the types that are defined for a repository, select **Type** from the **View** pull-down in the content management section of the Portal Administration Portal.

Figure 2



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 want to add to the repository. The 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

- At least one of the primary properties of a content type MUST be binary.
- 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 respository, 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

Lifecyclestatus	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.

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.

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 `-Xmsxxxm` as a parameter to the BulkLoader command, where *xxx* is the number of megabytes. For example `-Xms1000m`.

The following script is provided with BEA Weblogic Workshop:

Windows: `Weblogic81b\portal\bin/load_cm_data.cmd`
Unix: `Weblogic81b/portal/bin/load_cm_data.sh`

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 explains how to set up and run this command script.

Note: BEA Weblogic server must be running when you use BulkLoader.

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\weblogic81
   ```

3. Set the `CM_DATA` variable to point to the parent directory of the directory containing the content you want 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. For example:

   ```
   %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 the following table:

Table 2

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 WebLogic Portal application with which the repository is running.
`-d`	Specifies the base directory that contains the directories and files you want 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.
`file1...filen`	Specifies 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 the 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.

```
%JAVA_HOME%\bin\java -classpath %CLASSPATH%
com.bea.content.loader.bulk.BulkLoader -verbose -repository
"MyRepository" -application portalApp -d D:\media Images Audio Doc
```

The following command loads all files in `D:\media\images`. The command does not recurse into subdirectories. Metadata files with a `*.info.properties` naming convention are recognized.

```
%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
`-repository <repository name>`	Specifies the name of the repository to run the loader against.
`-application <app name>`	Specifies the name of the application to run the loader against.
`-url <wls url>`	Specifies the WebLogic Server instance host where the content manager is running.
`-user <principal username>`	Specifies the username for the principal permitted to access the Loader EJB resource.
`-password <principal password>`	Specifies the password for the principal permitted to access the LoaderEJB resource.
`-d <dir>`	Specifies the base directory that contains the directories and files you want 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 4 Optional BulkLoader parameters

Optional Parameters	Description
`-recurse`	Recurse into directories. (Default)
`+recurse`	Do not recurse into directories.

Optional Parameters	Description
-metaparse	Parses HTML files for META tags. (Default)
+metaparse	Does not parse HTML files for META tags.
-hidden	Ignores hidden files and directories. (Default)
+hidden	Includes hidden files and directories.
-inheritProps	Inherits metadata properties when recursing. (Default)
+inheritProps	Does not inherit metadata properties when recursing.
-ignoreErrors	Ignores errors while loading content (errors are still reported).
+ignoreErrors	Stops processing 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.
-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 filename extension for metadata property files. The value should start with a ".". This defaults to .md.properties.

Optional Parameters	Description
`-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.
`--`	Considers 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`	Displays command line usage.

Example BulkLoader Script

The following script is provided with WebLogic Workshop. The script configures the appropriate paths and runs the BulkLoader program. You can modify this script as you want, to suit your specific environment and needs.

Listing 1-1 BulkLoader Script (Windows): `Weblogic81b/portal/bin/load_cm_data.cmd`

```
@ECHO OFF
REM
#############################################################################
REM #      (c) BEA SYSTEMS INC. All rights reserved
REM #
REM
#############################################################################
SETLOCAL
SET PLATFORM_HOME=C:\bea\weblogic81
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
```