Oracle® WebCenter Sites

Administrator's Guide for Content Integration Platform for EMC Documentum

11*g* Release 1 (11.1.1)

February 2012

ORACLE®

Oracle® WebCenter Sites Administrator's Guide for Content Integration Platform for EMC Documentum, 11*g* Release 1 (11.1.1)

Primary Author: Tatiana Kolubayev

Contributor: Valentin Vakar, Chandrashekar Avadhani, Suman Saha

Table of

Contents

About This Guide

This guide contains procedures for installing, configuring, and using Oracle WebCenter Sites: Content Integration Platform for EMC Documentum to transfer content between Oracle WebCenter Sites and EMC Documentum. Supported modes include publishing Documentum objects to WebCenter Sites and archiving the assets of WebCenter Sites to EMC Documentum.

Applications discussed in this guide are former FatWire products. Naming conventions are the following:

- *Oracle WebCenter Sites* is the current name of the product previously known as *FatWire Content Server.* In this guide, *Oracle WebCenter Sites* is also called *WebCenter Sites*.
- *Oracle WebCenter Sites: Content Integration Platform* is the current name of the application previously known as *FatWire Content Integration Platform.* In this guide, *Oracle WebCenter Sites*: *Content Integration Platform* is also called *Content Integration Platform, or CIP.* The version described in this guide supports integration with EMC Documentum.
- *Oracle WebCenter Sites: Web Experience Management Framework* is the current name of the environment previously known as *FatWire Web Experience Management Framework.* In this guide, *Oracle WebCenter Sites: Web Experience Management Framework* is also called *WEM Framework.*
- *Oracle WebCenter Sites: Community* is the current name of the application previously known as *FatWire Community Server.* In this guide, *Oracle WebCenter Sites: Community* is also called *Community.*

Content Integration Platform for EMC Documentum integrates with Oracle WebCenter Sites according to specifications in the *Oracle WebCenter Sites 11g Release 1 (11.1.1.x) Certification Matrix*. For additional information, see the release notes for Content Integration Platform for EMC Documentum. Check the WebCenter Sites documentation site regularly for updates to the *Certification Matrix* and release notes.

Audience

This guide is intended for general administrators of WebCenter Sites, who also have experience with installing and configuring enterprise-level software. Also required is a

strong understanding of the WebCenter Sites flex asset model and RealTime publishing, in addition to a clear understanding of EMC Documentum and its data models.

Related Documents

For more information, see the following documents:

- *Oracle WebCenter Sites Administrator's Guide*
- *Oracle WebCenter Sites: Developing a Java Adapter and Plug-In for Content Integration Platform*

Conventions

The following text conventions are used in this guide:

- **Boldface** type indicates graphical user interface elements that you select.
- *Italic* type indicates book titles, emphasis, or variables for which you supply particular values.
- `Monospace` type indicates file names, URLs, sample code, or text that appears on the screen.
- **`Monospace bold`** type indicates a command.

Terms and Acronyms

This guide uses CIP-specific terms such as *CS Data Store*. The terms are defined in the "Glossary" section at the end of this guide.

Third-Party Libraries

Oracle WebCenter Sites and its applications include third-party libraries. For additional information, see *Oracle WebCenter Sites 11gR1: Third-Party Licenses*.

Chapter 1

Installing Content Integration Platform for EMC Documentum

This chapter contains procedures for installing and configuring the Oracle WebCenter Sites: Content Integration Platform for EMC Documentum to support content transfer between WebCenter Sites and Documentum installations.

This chapter contains the following sections:

Installation Overview

Content Integration Platform for EMC Documentum enables bidirectional communication between WebCenter Sites and EMC Documentum:

- Publishing EMC Documentum objects to WebCenter Sites.
- Archiving WebCenter Sites assets on EMC Documentum.

  In the archival process, the CS DataStore, rather than WebCenter Sites, is treated by CIP as the source of data. The CS DataStore is created when assets are published from WebCenter Sites to a RealTime destination with archiving enabled. The CS DataStore stores the published assets in its file system.

CIP supports publishing and archiving by replicating the source system's metadata to the target system. Schema that stores the source system's content is thus established on the target system. Content on the source system can then be transferred to the target system via the CIP command-line interface and **`cipcommander publish`** command. Once the publishing session ends, the synchronization engine starts monitoring the source system's published (or archived) workspaces for changes to content and automatically replicates the modified content to the target system.

This chapter shows you how to install CIP components to support publishing, archiving, or both processes, depending on your requirements:

- To support publishing or archiving, you must install Content Integration Agent, which manages metadata and the associated content. Content Integration Agent includes:
  - `mappings.xml`, which maps the source system's metadata to the target system.
  - `cipcommander`, a command-line application used to run the CIP `publish` command. The same command publishes to WebCenter Sites (via Sites Agent Services) and archives to Documentum.
  - synchronization engine, which monitors the source's published (or archived) workspaces in order to keep source and target systems in agreement.
  - `catalog.xml`, which stores information about published and archived items and serves as a configuration file for tuning the synchronization process.
- To enable publishing, you will install a programmatic publishing interface and database tables to store published objects, including their object type definitions:
  - The programmatic interface, named "Sites Agent Services," receives Documentum data from Content Integration Agent and stores the data to WebCenter Sites' database tables. The stored data consists of metadata in WebCenter Sites-compliant format and objects associated with the metadata.
  - Database tables for storing published objects are in the form of the `Documentum` flex family, a data model that must be installed on the WebCenter Sites management system and enabled on one of its content management sites. By default, the `Documentum` flex family also contains definitions of the Documentum object types `dm_folder` and `dm_document`. Both object types are mapped in the default `mappings.xml` file.
- To enable archiving, you will install the `fw_asset` and `fw_document` object types on Documentum to provide the schema for storing WebCenter Sites assets as Documentum objects. The object types are mapped in `mappings.xml`.

Prerequisites

Before installing Content Integration Platform, do the following:

- Download the *Oracle WebCenter Sites Certification Matrix* and release notes for this content integration product.
- Download the following:
  - Microsoft Visual C++ 2008 Redistributable (x86) from `http://www.microsoft.com`
  - OpenSSL from `http://www.openssl.org`
- Read this chapter to gain an understanding of the installation process and note the information you will need to provide.
- If you wish to set up CIP for publishing operations, create or select a content management site on the WebCenter Sites staging system for the `Documentum` flex family you will be installing.
- In archiving operations, CIP uses the WebCenter Sites: Web Experience Management (WEM) Framework.
- At the end of the CIP installation process, you have the option to configure event notification. If you choose to configure event notification, you must do so before publishing or archiving.

Also consider the following post-installation scenario: If your WebCenter Sites delivery system is running the Oracle WebCenter Sites: Community application and you wish to archive site visitors' comments and reviews, you will need to RealTime publish the comments and reviews from the delivery system to a separate WebCenter Sites system. For more information, see "Step VI. Archive Visitor-Generated Content," on page 41.

Packaging

Content Integration Platform is delivered as the following set of files:

File	Description
`cipagent-`*`vNo`*`.msi` (for Windows) `cipagent-`*`vNo`*`.rpm.bin` (for Linux)	For publishing and archiving. These files install Content Integration Agent (`cipcommander` and `service`) and configuration files that control the Content Integration Agent's process. One of these files must be installed on a machine with the supported operating system and the ability to access both the source and target systems.
`csagentservices.war`	For publishing. This file installs Sites Agent Services, including property files for setting the detail of log files and regulating access to the WebCenter Sites database. This file must be deployed on a system other than delivery. The system must have access to the WebCenter Sites `Shared` directory.

File	Description
`cs_documentum_schema.zip`	For publishing. This file installs the `Documentum` flex family on WebCenter Sites, on the CM site that you specify.
`documentum_cs_schema.dar`	For archiving. This file installs schema on the Documentum system to support the content of the CS DataStore.

Installation Steps

In this section, you will install Content Integration Platform for EMC Documentum. Your steps are the following:

Step I. Install Content Integration Agent

1. If you are using Windows, install Microsoft Visual C++ 2008 Redistributable Package (x86) on the machine that will host Content Integration Agent.
2. Run the `cipagent` file on a machine with a supported operating system and the ability to access the source and target systems.
   - **Windows:**

     Run `cipagent-2.0.0-127.msi` and follow the prompts.

     The following folders are created in the target directory:

     ```
     bin
       cipagent.exe
       cipcommander.exe
     conf
       .. all conf files
     security
        .. all certificates and private keys
     logs
       ..log file
     licenses
       ..licenses
     ```

   - **Linux:**

     Run (as a root user) the following command on the source system:

     `./cipagent-2.0.0-128.el5.i386.rpm.bin`

     The following directories are installed:

     ```
     usr
       local
         bin
           cipagent -exe
     ```

```
        cipcommander
      lib
        cipagent
          ..all libraries
share
  cipagent
    conf
    security
    logs
    licenses
```

3. Back up the configuration file `catalog.xml` (located in `integration_agent/conf/`).
4. Edit `catalog.xml`.

   The `catalog.xml` file stores configuration settings that Content Integration Agent requires to connect to the source system and Oracle WebCenter Sites. You will edit this file to provide Content Integration Agent with system location and user information.

   a. Open `catalog.xml` in a text editor.

   b. Edit the adapter for Oracle WebCenter Sites:

      Locate the provider element with name "`cs`" and id "`70b1e307-26a1-499c-9295-cf0b6bd01342`" and set the following parameters:

      - **urlAS**: Point to the Web Services module deployed with WebCenter Sites. Only the host name and port need to be modified. Typically, they name the host and the port number where WebCenter Sites is running. Do not alter the context name and context-related path unless you they differ from the default (`http://localhost:8080/csagentservices/InfostoriaService`).
      - **username**: User name of the account that has permissions to modify WebCenter Sites database tables (e.g., `fwadmin`, the general administrator).
      - **password**: Above user's password (e.g., `xceladmin`, assuming `fwadmin` is the username).
      - **context**: Leave this blank.

   c. Edit the adapter for the EMC Documentum installation:

      Locate the provider element with name "`documentum`" and id "`d7a96a63-e78c-407c-8d7f-e84988806e49`" and set the following parameters:

      - **urlDocumentum:** URL pointing to the server where Documentum Foundation Services (DFS) are running. Include the context name, but omit context-related paths. Typically you need to modify only the host name (the default value is `http://localhost:9090`).
      - **username**: User name for the account that has permissions to publishable content.
      - **password**: Above user's password.
      - **filter:** This parameter contains a `where` clause which is applied to filter out document versions that are not intended for publication. For example, to publish the latest approved version of a document, you can specify values such as the following:

        ```
        r_policy_id = '0000000000000000'
        ```

- or -

`r_current_state = 1`

(taking into account that the approved state has `"1"` as an index)

d. Save `catalog.xml`.

5. Restart the Content Integration Agent executable:
   - **Windows:** Restart the Content Integration Agent service.
   - **Linux:** Type as root user: **`/sbin/service restart cipagent`**

> **Note**
>
> The Content Integration Agent executable can be run as a standalone process or as a system daemon. The executable will start a simple HTTP server on the default port `7070`, which is reserved for `CIPCommander` communications with Content Integration Agent. Port `7070` is bound to the localhost, and therefore does not expose your system to any additional security risks.
>
> The `fileserver` facility default configuration takes port 7071 and attempts to automatically detect the host name. If multiple network interfaces are installed on the machine where Agent is running, we advise changing `auto` to the DNS name or the IP address that is accessible from the Sites Agent Services installation.
>
> Should you need to change the port, edit the port designation in `facilities.xml` and add **`-p <port>`** to all commands that start `CIPCommander`.

6. In `facilities.xml`, enable the java facility section. Specify the path to Java on your system.
7. Continue to the next step, "A. Installing Sites Agent Services."

Step II. Install CIP Publishing Components

If you wish to install only archival capability, skip to "Step III. Install Schema to Support Archiving to Documentum," on page 14. Otherwise, complete all the steps in this section to install publishing capability.

A. Installing Sites Agent Services

> **Note**
>
> Sites Agent Services can be installed on any WebCenter Sites system other than delivery. We recommend a content management (staging) system.

1. Edit the following files in `csagentservices.war` (all the files are located in `csagentservices/WEB-INF/classes`):
   - `commons-logging.properties`: defines the log file and log detail settings.
   - `csAgentServices.properties`: enables access to the WebCenter Sites database.

a. Using a text editor, edit `commons-logging.properties` to point to the Sites Agent Services log file (`agentservices.log`).

b. Create a data source specific to the application server. (More information is available in the guide for installing WebCenter Sites on the application server you are using.)

c. Modify `csAgentServices.properties` to enable access to the WebCenter Sites database.

   1) Using a text editor, set the following properties:

      - **`uploader.username`:** User name of an account with permissions to edit flex families.
      - **`uploader.password`:** Password for the above user name.
      - **`cs.installDir`**: WebCenter Sites installation directory (e.g., `C:\CS`)
      - **cs.url**: WebCenter Sites URL. Point to the WebCenter Sites web application. The default value is: `http://localhost:8080/cs`

   2) Save `csAgentServices.properties`.

2. Deploy `csagentservices.war` on the application server on the WebCenter Sites' host.
3. Restart the application server.
4. Continue to the next step, "B. Installing EMC Documentum Schema on Oracle WebCenter Sites."

B. Installing EMC Documentum Schema on Oracle WebCenter Sites

In this step, you will import `cs_documentum_schema.zip` into WebCenter Sites to support the publication of Documentum objects to WebCenter Sites.

To install the Documentum schema

1. Run `catalogmover.bat` (or `catalogmover.sh` on Linux) from the WebCenter Sites installation directory.

Note

Before using CatalogMover, connect it to WebCenter Sites:

1. Choose **Server** > **Connect**.
2. Provide the following information:
   - Server: The name of the HTTP server you want to connect to, and the port on which the server is running.
   - Name: **`ContentServer`**
   - Password: **`<password>`**
   - Below the "Password" field, select (or enter) a value that applies to your WebCenter Sites installation.
3. Click **Connect**.

2. Go to **Catalog** > **Auto Import Catalog(s)**.
   a. Select the file to import.
   b. In the import dialog, fill in the fields as shown below:

      Catalog Data Directory: Leave the default value

      Catalog ACL List: Browser,SiteGod,xceleditor,xceladmin
3. Log in to the WebCenter Sites Admin interface as a general administrator (**fwadmin / xceladmin**, by default) and continue as follows:
   a. Enable the `Documentum` flex family on the content management site you have chosen or created, as explained in "Prerequisites," on page 9.
   b. For quick access to published content, create a tree tab (named **Documentum**, for example).

   For instructions on enabling flex families, creating sites, and creating tree tabs, see the *Oracle WebCenter Sites Administrator's Guide*.
4. Continue to the next step, "Step III. Install Schema to Support Archiving to Documentum."

Step III. Install Schema to Support Archiving to Documentum

Install `documentum_cs_schema.dar` using Documentum Composer (installation instructions and user information are available in the *Documentum Composer User Guide):*

```
CREATE TYPE fw_asset (
fw_id char(255),
fw_name char(64),
fw_createdby char(64),
fw_createddate time,
fw_description char(128),
fw_publist char(255),
fw_status char(2),
fw_subtype char(32),
fw_updatedby char(64),
fw_updateddate time,
fw_publisheddate time
) WITH SUPERTYPE dm_document

CREATE TYPE fw_document (
fw_publisheddate time,
fw_name char(64)
) WITH SUPERTYPE dm_document
```

Step IV. Back Up the Default mappings.xml File

The `mappings.xml` file is located on the server that hosts Content Integration Agent.

Next Step

Once CIP is installed, complete the following steps, as necessary:

- **Optional**. Configure event notification workflows to inform CIP administrators about events that are triggered on the target system by changes to data in the source system's monitored workspaces (i.e., published or archived workspaces).

  Note

  If you choose to configure event notification workflows, you must do so before publishing any Documentum objects or archiving any WebCenter Sites assets. For instructions on configuring event notification workflows, see Chapter 2, "Configuring Event Notification."

- To publish from Documentum to WebCenter Sites, skip to Chapter 3, "Publishing to Oracle WebCenter Sites" for background information and instructions.
- To archive WebCenter Sites assets to Documentum, skip to Chapter 4, "Archiving to EMC Documentum" for background information and instructions.

Chapter 2

Configuring Event Notification

This chapter contains the following sections:

Overview of Event Notification

Event notification keeps Content Integration Platform administrators informed about the synchronicity of source and target systems as changes to content are made on either system. Event-driven notices are delivered to administrators in a simple workflow process which can be enabled globally or selectively for specific events (such as item creation) and for specific types of items.

Note

If you choose to configure CIP for event notification, you must do so before publishing objects to WebCenter Sites or archiving assets to Documentum. For instructions, see:

- "Configuring Event Notification Workflows for WebCenter Sites"
- "Configuring Event Notification Workflows for EMC Documentum," on page 21

Configuring Event Notification Workflows for WebCenter Sites

In the CIP publishing model, event notification is the process of informing CIP administrators of actions that occur on WebCenter Sites when changes are made to monitored workspaces – cabinets or folders – on Documentum.

Monitoring of a workspace begins once its metadata and associated objects are first published to WebCenter Sites via the **`cipcommander publish`** command. The synchronization engine starts listening to the workspace for changes to content and automatically replicates them to WebCenter Sites, where they are treated as events. If event notification is configured, the events invoke WebCenter Sites workflows to send notices confirming the events. Failed events also trigger notices if the associated workflows are configured. Table 1 lists the supported events and default workflows.

Table 1: Default workflows for WebCenter Sites

Event in WebCenter Sites	Workflow Process
Asset creation	`CIPAssetCreated`. Invoked when an object is created in a monitored cabinet or folder and the counterpart asset is created in WebCenter Sites.
Asset deletion	`CIPAssetDeleted`. Invoked when an object is deleted from a monitored cabinet or folder and the counterpart asset is deleted from WebCenter Sites.
Asset deletion failure	`CIPAssetDeletionFailed`. Invoked when: • An object that was deleted from the monitored cabinet or folder is checked out on the WebCenter Sites system. • An object that was deleted from the monitored cabinet or folder has dependencies that would become unresolved on the WebCenter Sites system if the counterpart asset were to be deleted.

Table 1: Default workflows for WebCenter Sites *(continued)*

Event in WebCenter Sites	Workflow Process
Asset modification	`CIPAssetModified`. Invoked when an object in the monitored cabinet or folder is modified and the counterpart asset is created in WebCenter Sites.
Asset modification failure	`CIPAssetModificationFailed`. Invoked when an object in the monitored cabinet or folder is modified, but its counterpart asset is checked out in WebCenter Sites.

Default workflows have the following properties:

- Default workflows consist of one state and two steps. When an event occurs, only the first step of the corresponding workflow is taken. If the option "Assign from list of participants" is chosen, all members with the selected roles are assigned the first task.
- All supported events trigger notices to users with role `CIPAdmin`. Tasks may or may not be assigned, depending on the event:
  - A task is assigned for the following events: creation, deletion failure, modification, and modification failure. The task is simply a way of notifying the user whether an event occurred or failed on WebCenter Sites. The task can be removed; there is no obligation to take a step.
  - For deletion events, no task is assigned (the asset no longer exists).

Note

Although workflows related to the publishing processes can be created, it is typically more convenient to use workflows that are provided with CIP. If you wish to create your own workflows, refer to instructions in the *Oracle WebCenter Sites Administrator's Guide*.

Installing Default Workflows on WebCenter Sites

1. Run `catalogmover.bat` (or `catalogmover.sh` on Linux) from the WebCenter Sites installation directory.
2. Go to **Catalog** > **Auto Import Catalog(s)**.
   a. Select `workflows.zip` (in the same directory or level as all `cs_*_schema.zip` files).
   b. In the import dialog, fill in the fields as shown below:

      Catalog Data Directory: Leave the default value

      Catalog ACL List: Browser,SiteGod,xceleditor,xceladmin
3. Create the default workflows by invoking the following URL:

   ```
   http://<host>:<port>/<context_path>/
      ContentServer?pagename=OpenMarket/Xcelerate/Installation/
      CIPCreateWorkflows&username=<username>&<password>=<password>
   ```

   where:

   - `host` is the address of the WebCenter Sites installation
   - `port` is the port of the WebCenter Sites installation
   - `context_path` is the context path where the WebCenter Sites web application is deployed
   - `username` is the WebCenter Sites administrator's user name
   - `password` is the WebCenter Sites administrator's password

   For example, the URL of the default configuration is:

   ```
   http://localhost:8080/cs/ContentServer?pagename=OpenMarket/
      Xcelerate/Installation/CIPCreateWorkflows&
      username=fwadmin&password=xceladmin
   ```

   When the workflows are created, the following message is displayed:

   "`Workflows for Content Integration Platform were created successfully`"

Verifying the Installation of Default Workflows

When the default workflows are created, associated items (such as roles and email objects) are also created in WebCenter Sites.

To verify default workflows and their associated items

1. Log in to the WebCenter Sites Admin interface as a general administrator (default credentials: `fwadmin` / `xceladmin`).
2. Verify that the following items have been created:
   - `CIPAdmin` role, which will be used as the management role in all CIP workflows. All users with the `CIPAdmin` role will be notified of all CIP events in the default workflows.
   - Workflow processes:
     CIP Asset Created, CIP Asset Deleted, CIP Asset Deletion Failed, CIP Asset Modified, and CIP Asset ModificationFailed

- Workflow states:
  CIP Asset Created, CIP Asset Deleted, CIP Asset Deletion Failed, CIP Asset Modified, and CIP Asset Modification Failed
- Workflow step action:
  CIP Asset Deleted, which results in an email notice to the CIP administrators.
- Email object:
  CIP Asset Event

Enabling Default Workflows

Default workflows are pre-configured in the default `mappings.xml` file. Each listed asset type contains a commented workflow configuration section.

To enable a CIP workflow

1. Open `mappings.xml` (on the Content Integration Agent host), go to the section `<mapping id= “documentum2cs”>` and uncomment the required workflows (below) for each asset type that must be enabled for event notification:

```
<param name="assetCreatedProcess">CIPAssetCreated</param>
<param name="assetModifiedProcess">CIPAssetModified</param>
<param name="assetDeletedProcess">CIPAssetDeleted</param>
<param name="assetDeletionFailedProcess">CIPAssetDeletionFailed
   </param>
<param
   name="assetModificationFailedProcess">CIPAssetModification
   Failed</param>
```

2. Assign the `CIPAdmin` role to CIP administrators. Verify that CIP administrators are able to receive email. For instructions, see the *Oracle WebCenter Sites Administrator's Guide.*
3. If a relatively large number of events occur on the source system, it is best to use workflow groups, as they allow you to resolve tasks in bulk. Workflow groups are not packaged by default. They must be created manually. For instructions, see the *Oracle WebCenter Sites Administrator's Guide.*

Note

If a workflow group has the name of the invoked workflow process, the workflow process will be automatically added to the group.

Configuring Event Notification Workflows for EMC Documentum

In the archival model, event notification is the process of informing CIP administrators of actions that occur in Documentum folders when changes are made to monitored CS DataStores.

Monitoring of a CS DataStore begins when its metadata and associated files are first archived to Documentum via the **`cipcommander publish`** command. The synchronization engine starts listening for changes to the content of the CS DataStore and

automatically replicates them to the Documentum target folders, where they are treated as events. If event notification is configured, the events invoke Documentum workflows to send notices confirming the events. Table 2 lists the supported events.

Table 2: Event Notification Workflows for the Archival Process

Event in Documentum	Workflow
Object creation	Invoked when a new asset is created in the CS DataStore and its counterpart object is created in Documentum.
Object modification	Invoked when an asset is modified in the CS DataStore and its counterpart object is created in Documentum.

Note

Event notification related to the archival process requires you to create or reuse Documentum workflows.

To enable notification workflows for archival events

1. Create your own workflow processes in Documentum or re-use workflow processes. Define the packages as necessary.
2. Open `mappings.xml` and go to `<mapping id= “csds2documentum”>`.
   a. Uncomment the required workflow processes and packages (below) for each asset type that must be enabled for event notification:

```
<param name="objectCreatedProcess">ProcessName</param>
<param name="objectModifiedProcess">ProcessName</param>
<param name="objectCreatedPackage">PackageName</param>
<param name="objectModifiedPackage">PackageName</param>
```

   b. Replace *ProcessName* with the name of the workflow process that should be automatically started.
   c. *PackageName* is an optional parameter. If *PackageName* is not specified, an asset will be placed into the first available package.

Chapter 3

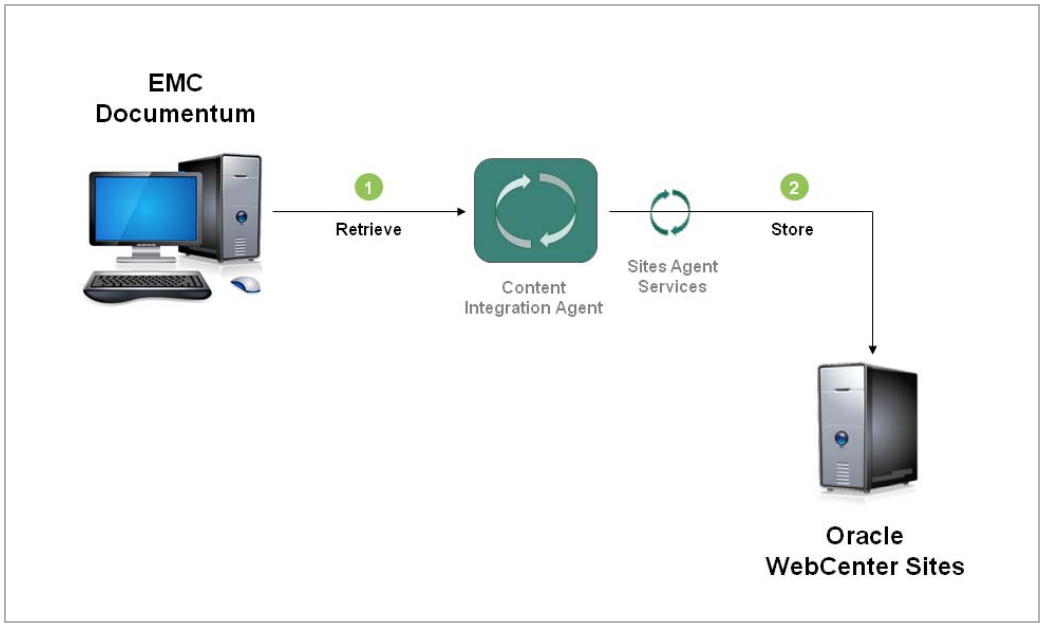
Publishing to Oracle WebCenter Sites

This chapter contains the following sections:

Overview of the Publishing Process

Publishing from EMC Documentum to Oracle WebCenter Sites requires the CIP components Content Integration Agent and Sites Agent Services, shown in Figure 1.

Figure 1: System Architecture for Publishing to Oracle WebCenter Sites



System Architecture and Process Flow

Content Integration Agent is used to synchronize the source and target workspaces via the **`cipcommander publish`** command, the synchronization engine, and the `mappings.xml` file, which provides the metadata map. Sites Agent Services exposes the web interface used by Content Integration Agent to perform the synchronization process. Following a publishing session, the synchronization process runs automatically. Details of the implementation are described below.

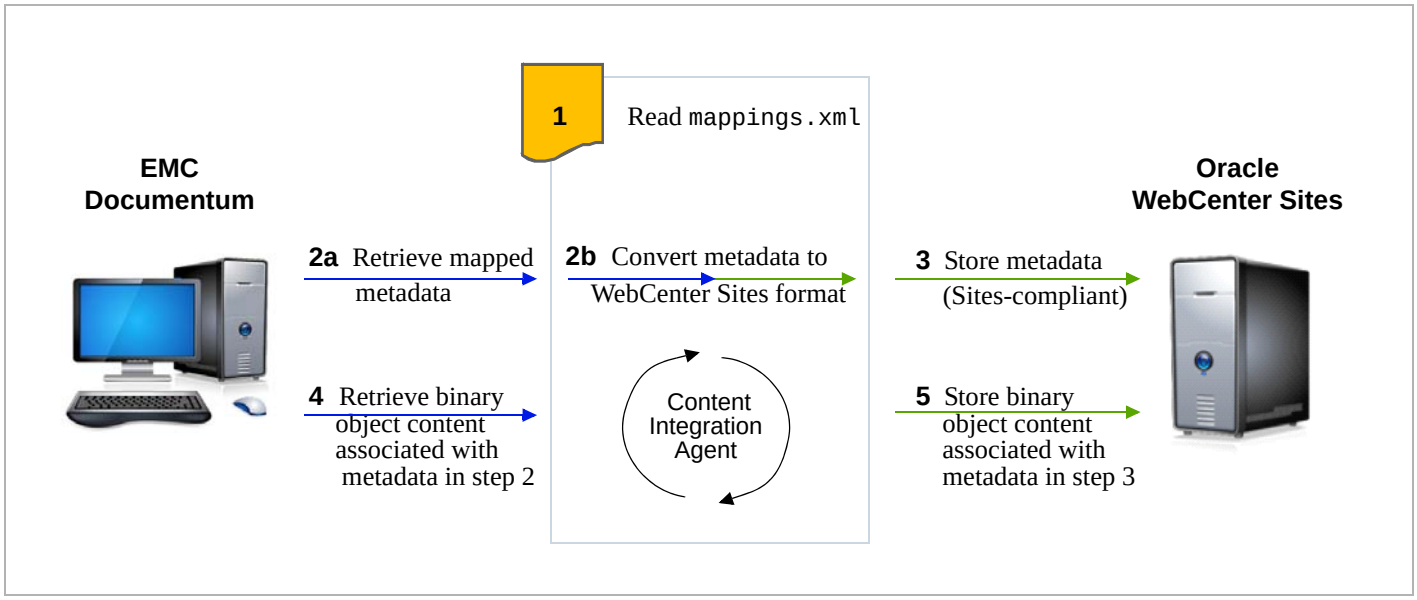
Initial Synchronization

When the **`cipcommander publish`** command is issued, the synchronization engine initializes the source and target by replicating metadata and associated content as outlined below (and in Figure 2, on page 25):

1. The synchronization engine reads the `mappings.xml` file,
2. refers to the Documentum workspace (i.e., cabinet or folder) named in the **`cipcommander publish`** command,
   a. reads the workspace's objects to retrieve their mapped metadata,
   b. converts the metadata to WebCenter Sites-compliant format, using the `mappings.xml` file
3. stores the WebCenter Sites-compliant metadata to WebCenter Sites (via Sites Agent Services),

4. retrieves from Documentum the content of binary objects associated with the metadata, and
5. stores the objects (via Sites Agent Services) to the target flex family.

Figure 2: Publishing Process



Monitoring the Data Source

Following the initial synchronization, the synchronization engine starts monitoring the published cabinet or folder and automatically replicates changes to WebCenter Sites (via Sites Agent Services). For example, when an object based on the published metadata is created, modified, or deleted in the monitored cabinet or folder, the synchronization engine replicates the new or modified object, or the object's deletion to WebCenter Sites. If metadata schema is modified, the workspace must be republished.

Tuning the Integration

Content Integration Agent contains a configuration file named `catalog.xml`, which stores information about the publishing session. Various parameters in the file can be tuned once the synchronization process is initialized. When objects are "unpublished," their information is deleted from `catalog.xml`. For more information about tuning, see "Tuning the Synchronization Process," on page 30.

Mapping Framework

The CIP mapping framework determines the success of the publishing and synchronization processes. Documentum objects can be published to WebCenter Sites as long as their metadata is mapped. The basic mapping framework for publishing involves two configurable components: the `Documentum` flex family and the `mappings.xml` file.

Target Flex Family

The `Documentum` flex family, provided with CIP, stores published objects and their object type definitions as WebCenter Sites assets. Appendix A provides flex family specifications.

For simplicity, we recommend using the default flex family. Should you need to create your own flex family, refer to instructions in the *Oracle WebCenter Sites Administrator's Guide*. Use the information in Appendix A of this guide as a model of the flex family.

mappings.xml

The default `mappings.xml` file contains a `documentum2cs` section, which specifies the mappings listed below:

- The `dm_folder` type maps to `Documentum_Folder;dm_folder` in the `Documentum` flex family, where
  - `Documentum_Folder` is a flex parent asset type that stores folder assets.
  - `dm_folder` is a parent definition (of type `Documentum Parent Definition`) that defines the folder type.
- The `dm_document` type maps to `Documentum_Document;dm_document` in the `Documentum` flex family, where
  - `Documentum_Document` is a flex asset type that stores document assets.
  - `dm_document` is a child definition (of type `Documentum Child Definition`) that defines the document type.
- Attributes are mapped in the `<descriptor-mapping .../>` tags. Attributes are named as listed under the "Types" node of the Documentum WebTop interface:
  - `title`
  - `subject`
  - `keywords`
  - `r_version_label`
  - `r_full_content_size`

Mapping, Publishing, and Synchronization

When publishing, bear in mind the following mapping specifications:

- Documentum objects based on default metadata can be published to WebCenter Sites without you having to modify either the default `mappings.xml` file or the `Documentum` flex family. Running the **`cipcommander publish`** command replicates the specified workspace and its subfolders to the `Documentum` flex family as *flex parents*; documents are replicated as *flex child assets*. The published workspace is then monitored by the synchronization engine.
- Publishing objects based on custom metadata (such as a new document type) requires you to first update at least the flex family with the new metadata. The `mappings.xml` file may or may not require updates, depending on the nature of the metadata. Details of custom mappings can be found in "Publishing via Customized Mappings," on page 29.

Publishing Procedures

Publishing via the Default Mapping

In this section, you will publish cabinets and folders to WebCenter Sites using the default `mappings.xml` file and `Documentum` flex family.

To publish

1. Start Content Integration Agent.

   Note

   If you changed the port in step 5 on page 12 (starting Content Integration Agent), make sure that the new port is set in `facilities.xml`, and add **`-p <port>`** to the **`cipcommander publish`** command in step 3, below (which starts `CIPCommander`).

2. Run the `CIPCommander` executable (located in the `bin` folder of the system where Content Integration Agent is installed).
3. Publish objects of the types that are specified in the default `mappings.xml` file (for definitions of publishing parameters, see Table 3, on page 28):

```
cipcommander
  publish <source_providerid> <target_providerid>
  -source_repname <cabinet_name>
  -source_path <path_in_cabinet>
  -target_repname <CS_content_management_site>
  -mapping <mapping_id>
  -replic_mode <full | new | updated>
  -bulk_resynch_interval <seconds>
```

Examples:

- To publish the `Images` cabinet to the “CIPDemo” content management site:

```
cipcommander publish d7a96a63-e78c-407c-8d7f-e84988806e49
    70b1e307-26a1-499c-9295-cf0b6bd01342
  -source_repname Images
  -source_path /
  -target_repname CIPDemo
  -mapping documentum2cs
```

- To publish the `/Sample/Trees` folder in the `Images` cabinet to the “CIPDemo” content management site:

```
cipcommander publish d7a96a63-e78c-407c-8d7f-e84988806e49
    70b1e307-26a1-499c-9295-cf0b6bd01342
  -source_repname Images
  -source_path /Sample/Trees
  -target_repname CIPDemo
  -mapping documentum2cs
```

4. When the publishing session ends, the synchronization engine starts monitoring the published object.
   - Verify that modifications and deletions are replicated to WebCenter Sites (for example, modify the replicated objects, add folders and documents to the monitored object, and delete documents).
   - Objects can also be "unpublished." For information, see "Unpublishing," on page 30.
   - To optimize the synchronization process, see "Tuning the Synchronization Process," on page 30.

Table 3: Publishing Parameters

Publishing Parameter	Required	Value
`<source_providerid>`	R	Provider ID for Documentum: `d7a96a63-e78c-407c-8d7f-e84988806e49`
`<target_providerid>`	R	Provider ID for WebCenter Sites: `70b1e307-26a1-499c-9295-cf0b6bd01342`
`-source_repname`	R	`<cabinet_name>`: Name of the cabinet containing the objects to be published. Enter the name exactly as it appears in the URL.
`-source_path`		`<path_in_cabinet>`: Path to the object you want to publish. • `/` (to publish the cabinet specified by `<source_repname>`) • `/<folder>/<folder>/... /<folder>` (to publish the last folder in the path)
`-target_repname`	R	Name of the content management site (on WebCenter Sites) on which the target flex family is enabled. Enter the site's display name, exactly as it appears on the **Admin** tab in the WebCenter Sites Admin interface.
`-mapping`	R	`<mapping_id>`: Value of the `mapping id` in `mappings.xml`. The value is `documentum2cs`.
`-replic_mode`		`full | new | updated` • `full` means that a full replication will be performed (by default), i.e., newly created items, updated items, and deletions. • `new` means that only newly created items will be replicated (updates and deletions will not be replicated). • `updated` means that only new and updated items will be replicated (deletions will not be replicated).
`-bulk_resynch_interval`		`<seconds>`: Number of seconds between two successive synchronization events. **Default value:** `600` For optimal performance, set the synchronization interval to a value that agrees with the frequency of updates to the monitored folders. For more information, see "Tuning the Synchronization Process," on page 30.

Publishing via Customized Mappings

If the objects you plan to publish are based on unmapped metadata, you must first map the object types.

To publish via customized mappings

1. Depending on which type of metadata you have created, update the relevant mapping components as shown below. (We suggest reusing the `Documentum` flex family. The `mappings.xml` file is located on the Content Integration Agent host).

New Metadata	Update	Guidelines
New folder type	`Documentum` flex family	Create a parent definition for the new folder type. (The default parent definition is `dm_folder`.)
	`mappings.xml`	Map the new folder type (`source id`). The `target id` takes the value `Documentum_Parent;<parent definition>` (where `Documentum_Parent` defines the storage table for folder assets).
New document type	`Documentum` flex family	Create a child definition for the new document type. (The default child definition is `dm_document`.)
	`mappings.xml`	Map the new document type (`source id`). The `target id` takes the value `Documentum_Child;<child definition>` (where `Documentum_Child` defines the storage table for document assets).
New document attribute	`Documentum` flex family	Add the new attribute to the Documentum flex family.
	Updating `mappings.xml` is conditional	Mapping the new attribute is required only if its name differs on the source and target. **Note:** Incorrect mapping of attributes does not stop the publication process, but it does produce a warning message and an entry in the log file.

2. If you create flex filters, add the corresponding `jar` files to both the WebCenter Sites and the Sites Agent Services applications.
3. Publish the objects. For instructions, see “Publishing Procedures,” on page 27.

Maintaining the Integrated Systems

Tuning the Synchronization Process

When a cabinet or folder is published, `catalog.xml` is updated with data points from the **`cipcommander publish`** command. The data points identify the Documentum and WebCenter Sites systems (in the `<workspace>` tags) and specify replication settings (in the `<replication>` tag).

Following a publishing session, the synchronization engine monitors the published cabinet (folder), using `catalog.xml`. The `BulkResynchInterval` and `ReplicMode` parameters listed in the sample file below can be reset as shown in Table 3, on page 28. (The `catalog.xml` file is located in the `conf` folder on the Content Integration Agent server.)

Sample catalog.xml

```
<workspace id="41ce3f11-0411-46cb-b974-429162249462">
  <provider-ref refid="d7a96a63-e78c-407c-8d7f-e84988806e49" />
  <init-params>
    <param name="repname">Documents</param>
    <param name="path">/Images</param>
    <param name="repid">0c000001800045fe</param>
    <param name="itemid">0b0000018002c58e</param>
  </init-params>
</workspace>
<workspace id="6af59904-c6a3-4588-af7b-76f1422d6c10">
  <provider-ref refid="70b1e307-26a1-499c-9295-cf0b6bd01342" />
  <init-params>
    <param name="repname">FirstSiteII</param>
    <param name="repid">68ef906a-6c59-406a-84c2-b73b098cdb93</param>
  </init-params>
</workspace>
<replication>
  <link id="332e73c8-e977-4ba4-85c2-d7636281e192">
    <source-ref refid="41ce3f11-0411-46cb-b974-429162249462" />
    <target-ref refid="6af59904-c6a3-4588-af7b-76f1422d6c10" />
    <mapping-ref refid="documentum2cs" />
    <init-params>
      <param name="BulkResynchInterval">600</param>
      <param name="ReplicMode">full</param>
      <param name="IncrementalSyncDelay">10</param>
    </init-params>
  </link>
</replication>
```

Unpublishing

You can unpublish objects from `catalog.xml` and WebCenter Sites by executing the **`cipcommander unpublish`** command. The command clears `catalog.xml` of all entries that are associated with published objects (for a sample publication entry, see the

sample file on page 30). The **`-delete`** parameter removes the same entries from WebCenter Sites' database.

The **`unpublish`** command takes the following form and parameters:

```
cipcommander unpublish <parameters>
```

Table 4: Unpublish Parameters

Unpublish Parameter	Description
`-all`	Clears `catalog.xml` of **all** publication entries.
`-linkid`	Clears `catalog.xml` of **selected** publication entries. **`linkid`** specifies the published object's link to the WebCenter Sites system. Use the value in the published object's <`link`> tag, which is nested within the object's <`replication`> tag (for sample code, see page 30). For example, to unpublish an object with `linkid` `332e73c8-e977-4ba4-85c2-d7636281e192` from `catalog.xml`, run the following command: `cipcommander unpublish linkid 332e73c8-e977-4ba4-85c2-d7636281e192`
`-delete`	Removes from WebCenter Sites' database the same objects that you are unpublishing from `catalog.xml`. **Legal values:** `<true | false>` **Default value:** `true`

Chapter 4

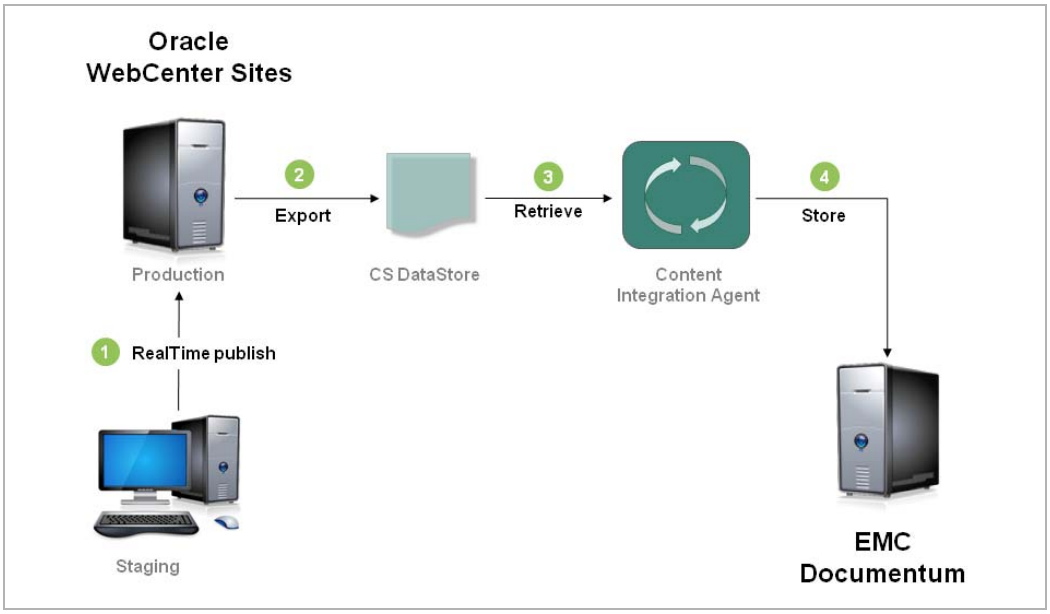
Archiving to EMC Documentum

This chapter contains the following sections:

Overview of the Archival Process

Archiving WebCenter Sites assets to EMC Documentum requires the components shown in Figure 3. Users who are familiar with the process of publishing Documentum objects to WebCenter Sites will recognize archiving to be a similar process.

Figure 3: System Architecture for Archiving



The main differences between publishing and archiving are the following:

- In the archival process, Content Integration Agent treats the CS DataStore, rather than WebCenter Sites, as its source of data. The CS DataStore contains assets in folders and files that map to Documentum folders of type `dm_folder` and files of type `fw_document`. The CS DataStore is created when assets in the WebCenter Sites system are published to a RealTime destination while archiving is enabled.
- Content Integration Agent archives data to Documentum directly. It does not require the Sites Agent Services component to store data on the target system.

System Architecture and Process Flow

Content Integration Agent synchronizes the CS DataStore and target Documentum folder via the `publish` command, the synchronization engine, and the `mappings.xml` file, which provides the metadata map. Following the initial archival session, the synchronization process runs automatically. Details of the implementation are described below.

Initial Synchronization

Issuing the `publish` command invokes the synchronization engine to archive the CS DataStore to Documentum and thereby initialize the synchronization process. The complete set of steps is outlined below (and shown in Figure 3):

1. Assets in WebCenter Sites are RealTime published with archiving enabled.

2. The CS DataStore is created.
3. Issuing the `publish` command invokes the synchronization engine, which then:
   a. refers to the CS DataStore (which is specified in the `publish` command),
   b. reads the files in the CS DataStore, retrieves their metadata, and
   c. converts the metadata to a Documentum-compliant format, using `mappings.xml`.
4. The synchronization engine then stores the CS DataStore files to the target Documentum folder.

Monitoring the CS DataStore

Following the initialization process, the synchronization engine monitors the archived CS DataStore and automatically replicates changes to the Documentum target folder. When an asset based on the archived metadata is created or modified in the monitored CS DataStore (during a RealTime publishing process), the synchronization engine replicates the new or modified asset to the target folder on Documentum. If metadata is modified, the `mappings.xml` file must be reconfigured and the CS DataStore must be republished.

Tuning the Integration

Content Integration Agent contains a configuration file named `catalog.xml`, which stores information about the archival session and allows tuning of the synchronization interval. For more information about tuning, see "Tuning the Synchronization Process," on page 42.

CS DataStore

The CS DataStore contains WebCenter Sites assets in folders and files that map to Documentum folders of type `dm_folder` and files of type `fw_document`.
A CS DataStore is created when assets are published to a RealTime destination with archiving enabled. The export path is:

```
<CS DataStore = cs.pgexportfolder>/CIP_DataStore/<CS DataStore for
   Publishing Destination>
```

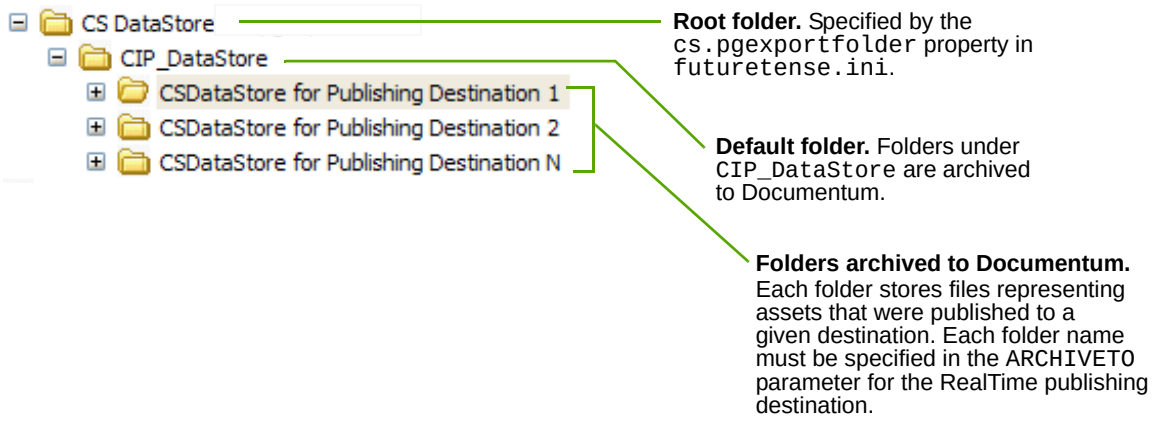


Figure 4, on page 36 illustrates the archival of a sample CS DataStore.

Figure 4: Sample CS DataStore Archived to Documentum

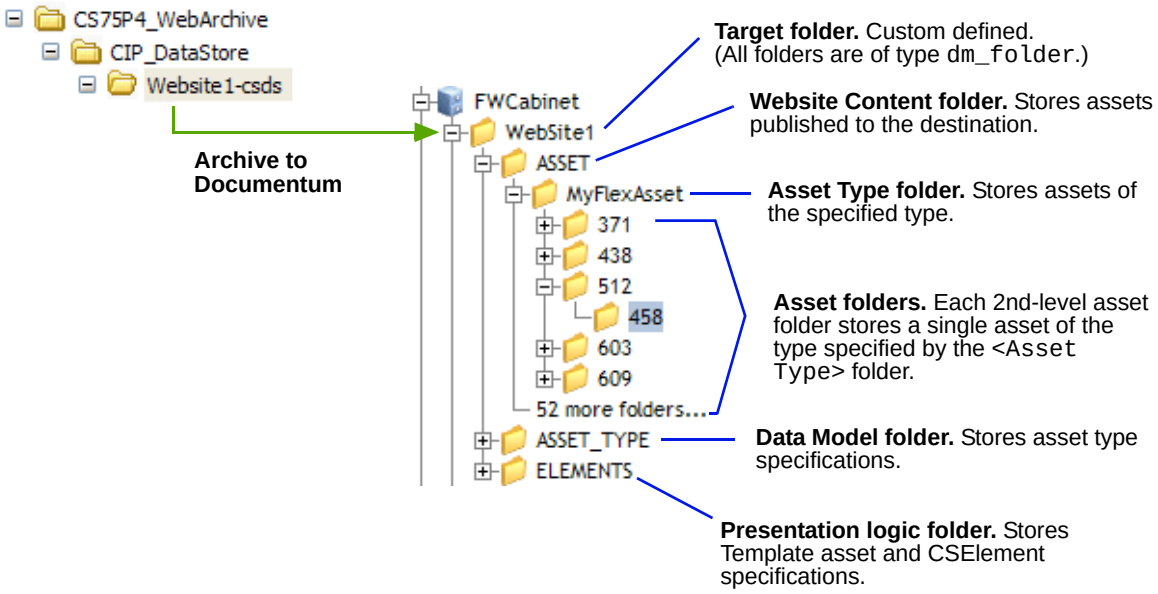
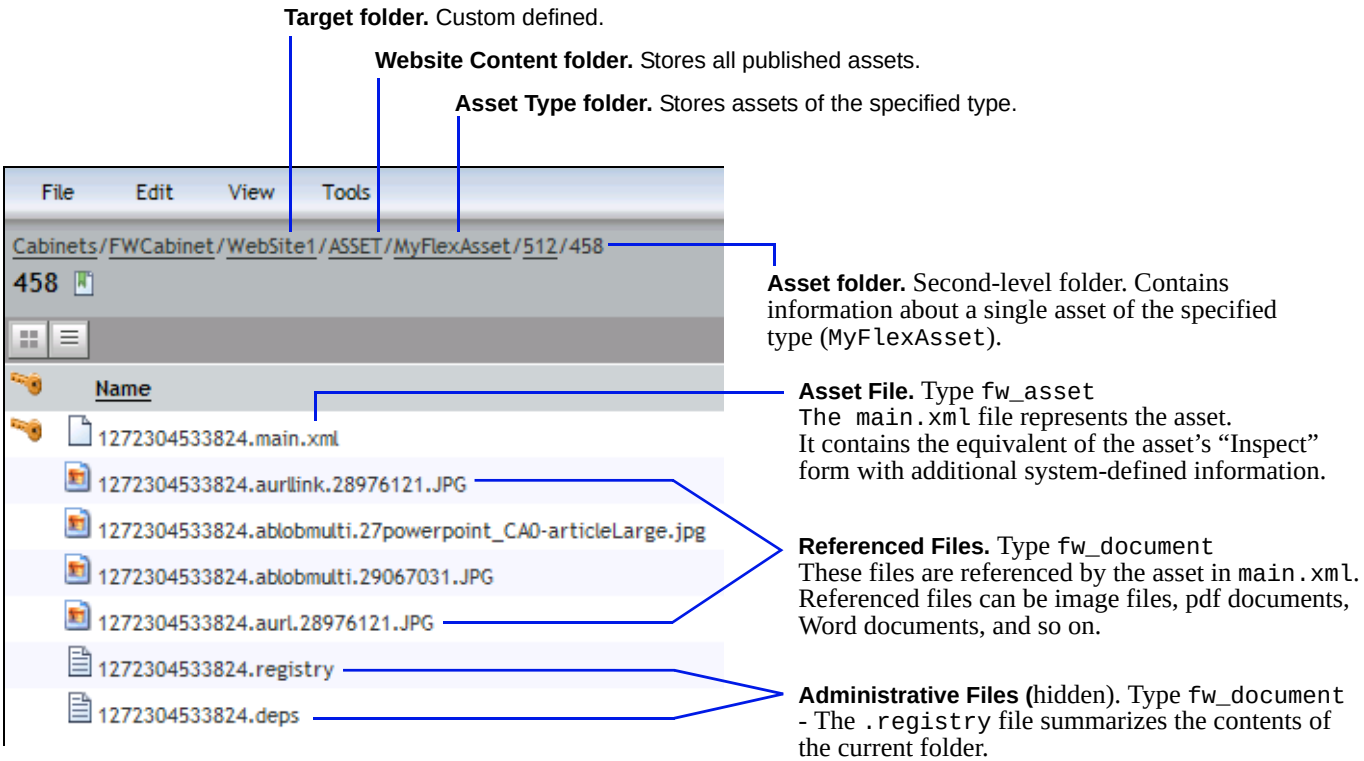


Figure 5 shows the contents of a typical asset folder (the same folder, named **458**, is also shown in Figure 4). **Note that the administrative files shown in Figure 5 are hidden files. They are shown here only for illustration purposes.**

Figure 5: Sample Asset Folder



Mapping Framework

The CIP mapping framework determines the success of the archival and synchronization processes. A CS DataStore can be archived to Documentum as long as its metadata is mapped. The basic mapping framework involves two configurable components: object types in the Documentum workspace and the `mappings.xml` file.

Object Types in the Documentum Workspace

Default Documentum target types are `fw_asset` and `fw_documentum`, which could be modified or replaced with custom types.

mappings.xml

The default `mappings.xml` file contains a `documentum2cs` section, which specifies the mappings listed below:

- `csds_Folder` is the data type for all folders in the CS DataStore. Folders of type `csds_Folder` map to Documentum folders of type `dm_folder`:

```
<assettype-mapping
    sourceid="csds_Folder" targetid="dm_folder"
    id="Folder" extends="Item" />
```

- `csds_Document` is the data type for all files (except `main.xml`) in the CS DataStore. Files of type `csds_Document` map to Documentum documents of type `fw_document`.

```
<assettype-mapping
    sourceid="csds_Document" targetid="fw_document"
    id="Document" extends="Item" />
```

- `csds_Asset` is the data type of the `main.xml` file, which defines the asset (see Figure 5, on page 36). Files of type `csds_Asset` map to Documentum documents of type `fw_asset`.

```
<assettype-mapping
    sourceid="csds_Asset" targetid="fw_asset"
    id="Asset" extends="Document">
```

- Attributes of the `csds_Asset` document type map to Documentum attributes as shown. All WebCenter Sites attributes are system-defined.

source id	target id	source id	target id
id	fw_id	publist	fw_publist
name	fw_name	status	fw_status
createdby	fw_createdby	subtype	fw_subtype
createddate	fw_createddate	updatedby	fw_updatedby
description	fw_description	updateddate	fw_updateddate

- The `datemodified` attribute is a special attribute that stores the last publication date. The date is taken from the last modification time of the corresponding asset file in the CS DataStore. The `datemodified` attribute maps to the Documentum attribute `fw_publisheddate`.

Mappings, Publishing, and Synchronization

When archiving, bear in mind the following mapping specifications:

- **Default mapping.** Any CS DataStore can be archived to Documentum without your having to modify the default `mappings.xml` file. Running the `publish` command archives the CS DataStore to the target Documentum folder. The source CS DataStore is then monitored by the synchronization engine. When a RealTime publishing session updates the CS DataStore, the new assets and modified assets are automatically replicated to the target Documentum folder by the synchronization engine.
- **Custom mapping.** You can map selected asset types and definitions to your own Documentum object type. Instructions are available in "Step III. Add Metadata to mappings.xml," on page 39.

Steps for Archiving WebCenter Sites Assets to Documentum

Step I. Prepare the Documentum System to Store WebCenter Sites Assets

1. Create a cabinet for WebCenter Sites assets (`FWCabinet`, for example).
2. Create a folder for the given assets (`WebSite1`, for example).
3. If you configured event notification, store the associated workflow processes anywhere on the Documentum system.

Step II. Configure the Path to the CS DataStore

In this step, you will enable the WebCenter Sites RealTime publishing system to export your selected site in a CS DataStore along the following path (see also "CS DataStore," on page 35):

```
<CS DataStore=cs.pgexportfolder>/CIP_DataStore/<CSDataStore for
   Publishing Destination>
```

where:

- `<cs.pgexportfolder>` defines the root directory of the CS DataStore. (The `cs.pgexportfolder` property is located in the `futuretense.ini` file on the WebCenter Sites delivery system.)
- `CIP_DataStore` is a default subdirectory of `<cs.pgexportfolder>`. Its subfolders will be archived on Documentum.

- `<CSDataStore for Publishing Destination>` is a subfolder that holds the assets of the published site.

To configure the path to the CS DataStore

1. Configure the root directory of the CS DataStore by setting the `cs.pgexportfolder` property in the `futuretense.ini` file of the delivery system.
2. Configure the RealTime publishing process to support archiving.
   a. Configure publishing as shown in the *Oracle WebCenter Sites Administrator's Guide*. If RealTime publishing is already configured, start with "Creating a RealTime Destination Definition on the Source System."
   b. In the "Add New Destination" form, set the "More Arguments" field as follows:

      ```
      ARCHIVETO=<CSDataStore for Publishing Destination>
      ```

      Note

      If you are publishing a small number of assets (less than thousands) and wish to store their files to the same folder in the CS DataStore, specify `USEHASHDIRS=false` to disable hash folders.

   c. Complete the remaining steps up to and including mirroring site configuration data to the destination database.

Step III. Add Metadata to mappings.xml

To add custom mappings to `mappings.xml`, include the following information, depending on whether you are mapping a flex or basic asset type:

For flex asset types, `sourceid` takes the form `<asset type>;<Definition>`. For basic asset types, `sourceid` takes the form `<asset type>`. The `targetid` specifies the corresponding Documentum object type.

For example:

- To map all flex assets of type `Content_C` with asset definition `FSII Article` to the `fw_content` object type, add the following line to `mappings.xml`:

  ```
  <assettype-mapping sourceid="Content_C;FSII Article"
     targetid="fw_content" id="Content">
  </assettype-mapping>
  ```

- To map all basic assets of type `FW_Article` to the `fw_article` object type, add the following line to `mappings.xml`:

  ```
  <assettype-mapping sourceid="FW_Article"
     targetid="fw_article" id="Article">
  </assettype-mapping>
  ```

Step IV. RealTime Publish the Site

Publish the content management site and verify that the CS DataStore was created in the specified path (in "Step II. Configure the Path to the CS DataStore").

Step V. Archive the CS DataStore on Documentum

In this step, you will run the CIP `publish` command to archive the `<CS DataStore for Publishing Destination>` and initialize the synchronization process.

Note

If you changed the port in step 5 on page 12 (starting Content Integration Agent), make sure that the new port is set in `facilities.xml`, and add **`-p <port>`** to the command in step 2, below (which starts `CIPCommander`).

1. Start Content Integration Agent.
2. Run the `CIPCommander` executable (located in the `bin` folder of the system where Content Integration Agent is installed):

```
cipcommander
   publish <source_providerid> <target_providerid>
   -source_repid <path to data store>
   -target_repname <cabinet name>
   -target_path <path within the cabinet>
   -mapping <mapping_id>
   -bulk_resynch_interval <seconds>
   -handlerset csdatastore
   -create false
   -replic_mode updated
```

For example:

```
cipcommander
  publish 7833d862-4f8b-4285-84f2-731d5af81865 d7a96a63-
  e78c-407c-8d7f-e84988806e49
  -source_repid c:\temp\csdatastore
  -target_repname Archive
  -target_path /fatwire
  -mapping csds2documentum
  -bulk_resynch_interval 60
  -handlerset csdatastore
  -create false
  -replic_mode updated
```

Table 5: Publishing Parameters

Publishing Parameter	Value
`<source_providerid>`	provider ID for the WebCenter Sites system: `7833d862-4f8b-4285-84f2-731d5af81865`
`<target_providerid>`	Provider ID for the Documentum system: `d7a96a63-e78c-407c-8d7f-e84988806e49`
`-source_repid`	`<path to data store>`: Path to the `<CS DataStore=cs.pgexportfolder>` folder on the file system.

Table 5: Publishing Parameters

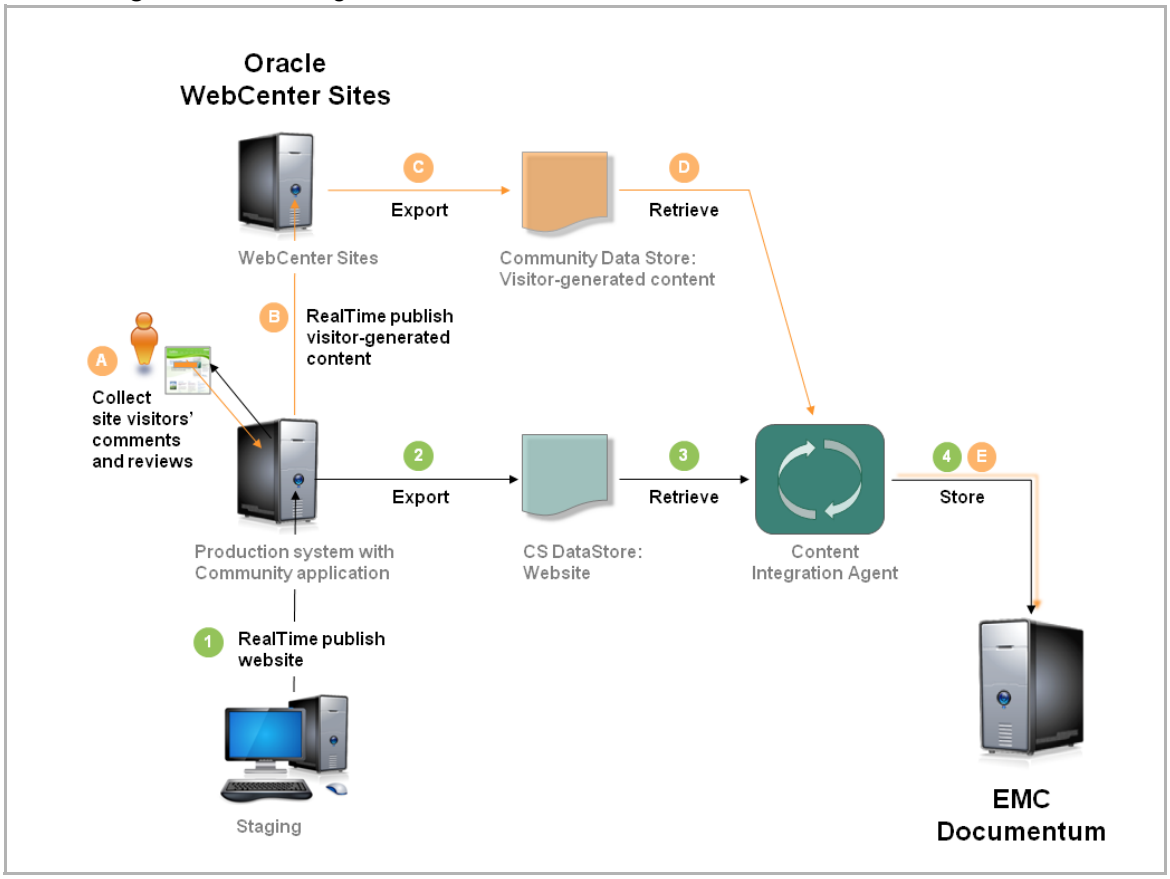
Publishing Parameter	Value
`-target_repname`	`<cabinet name>`: Name of the Documentum cabinet to which `<CSDataStore for Publishing Destination>` will be archived.
`-target_path`	`<path within the cabinet>`: Path to the Documentum folder in the cabinet specified by `target_repname`. To publish to the cabinet itself, skip this parameter.
`-mapping`	`<mapping_id>`: mapping identifier from the `mappings.xml` file. **Default value**: `csds2documentum`
`-bulk_resynch_interval`	`<seconds>`: Number of seconds between two successive synchronization events. For optimal performance, set this value to a number that correlates with the frequency of RealTime publishing sessions. **Default value:** `600`
`-handlerset`	References the handlerset element from `handlers.xml`. **Allowed value:** `csdatastore`
`-create`	Specifies whether to create target repository. **Allowed value:** `false`
`-replic_mode`	Specifies which types of changes will be replicated. **Allowed value:** `updated` (only new and updated items will be replicated; deletions will not be replicated).

3. Verify on Documentum the directory structure of the archived assets. For background information, see “CS DataStore,” on page 35.

Step VI. Archive Visitor-Generated Content

If your WebCenter Sites delivery system runs the WebCenter Sites: Community application and you wish to archive its visitor-generated content (comments and reviews), publish the content from the delivery system to a RealTime destination on a separate WebCenter Sites system (see Figure 6, on page 42). Procedures for archiving visitor-generated content are identical to those for archiving assets.

Figure 6: Archiving Visitor-Generated Content



Testing Synchronization

After the `publish` command executes, the archive session ends and the synchronization engine starts monitoring the source CS DataStore. Verify that modifications are replicated to Documentum:

1. Create and modify assets on the published content management site.
2. Republish the site to export the same `<CS DataStore for Publishing Destination>` folder.
3. Look for updates in the target Documentum folder.

Tuning the Synchronization Process

When the `publish` command executes, the CS DataStore is archived to Documentum and `catalog.xml` is updated with data points from the `publish` command. The data points identify the CS DataStore and Documentum system (in the `<workspace>` tags) and specify replication settings for the CS DataStore (in the `<replication>` tag).

Following the archival session, the synchronization engine starts monitoring the published CS DataStore. The synchronization interval can be reset in `catalog.xml`.

See `BulkResynchInterval` in Table 5, on page 40. (The `catalog.xml` file is located in the `conf` folder.)

Sample catalog.xml

```
<workspace id="776a0536-1af8-4e10-9b55-ecb9cfd715b8">
  <provider-ref refid="7833d862-4f8b-4285-84f2-731d5af81865" />
  <init-params>
    <param name="repid">C:\cs\export\CIP_datastore\DCTM</param>
    <param name="repname"></param>
  </init-params>
</workspace>
<workspace id="ae679aa2-572c-492a-b553-b7a866b60a2f">
  <provider-ref refid="d7a96a63-e78c-407c-8d7f-e84988806e49" />
  <init-params>
    <param name="repname">Archive</param>
    <param name="path">/FirstSiteII</param>
    <param name="repid">0c0000018002bd87</param>
    <param name="itemid">0b0000018002bd91</param>
  </init-params>
</workspace>
<replication>
  <link id="b5be4bc4-a8dc-4928-b3df-e0ac2851f4e4">
    <source-ref refid="776a0536-1af8-4e10-9b55-ecb9cfd715b8" />
    <target-ref refid="ae679aa2-572c-492a-b553-b7a866b60a2f" />
    <mapping-ref refid="csds2documentum" />
    <handlerset-ref refid="csdatastore" />
    <init-params>
      <param name="BulkResynchInterval">3</param>
      <param name="ReplicMode">updated</param>
      <param name="IncrementalSyncDelay">10</param>
    </init-params>
  </link>
</replication>
```

Appendix A

Default Mapping Specifications for Publishing

This appendix contains the following sections:

Mapping Framework

The default mapping framework for the publishing model supplies the following components (also listed in Table A-1):

- The `Documentum` flex family, pre-configured to match the object types and attributes listed above.
- A `mappings.xml` file, in which the object types and attributes listed above are mapped to assets in the `Documentum` flex family:
  - The `dm_folder` type is mapped to a flex parent definition asset named `dm_folder`.
  - The `dm_document` type is mapped to a flex definition asset named `dm_document`.
  - Attributes are mapped to flex assets of type `Documentum Attribute`.

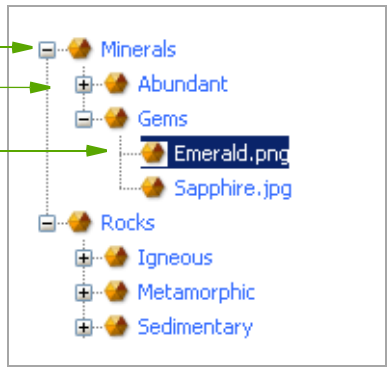
When the `publish` command is issued:

- Folders are published as flex parent assets to the `Documentum Folder` asset type.
- Documents are published as flex assets to the `Documentum Document` asset type.

During publishing, Content Integration Agent refers to the `mappings.xml` file to determine the types of objects to publish. The folder that is named in the **`publish`** command is the starting point of the publication process. The folder is published as a flex parent asset of type `Documentum Folder`, along with all the subfolders and documents it contains.

To reproduce the folder's structure (subfolders and documents), Content Integration Agent refers to path information. If subfolders exist, Content Integration Agent chains the counterpart `Documentum Folder` assets to reproduce the hierarchy.

Documents are treated as flex assets of type `Documentum Document`. They are placed under their respective `Documentum Folder` parents.



'Documentum' Flex Family Specifications

Table A-1 summarizes the mapping of default object types in Documentum to asset types and assets in WebCenter Sites' `Documentum` flex family. In customized implementations, you can either re-use the flex family or create your own.

Table A-1: Documentum Default Data and Flex Family Analogs

Documentum Object Type	WebCenter Sites: `Documentum` Flex Family	
	Flex Asset Type	**Assets**
Document Attribute[a] • **`title`** • **`subject`** • **`keywords`** • `r_version_label` • `r_full_content_size`	**Documentum Attribute** Stores document attributes.	• `title` • `subject` • `keywords` • `version label` • `file size`
Folder `dm_folder` (default)	**Documentum Parent Definition** Stores folder type definitions.	**Parent definitions** `dm_folder` (default)
	`Documentum_Folder` Flex Parent asset type. Stores folder assets.	Documentum folders
Document `dm_document` (default)	**Documentum Child Definition** Stores document type definitions.	**Document definitions** `dm_document` (default)
	`Documentum_Document` Flex (Child) asset type. Stores document assets.	Documentum documents
default mappings.xml: <assettype mapping>		
sourceid	**targetid**	
`dm_folder`	`Documentum_Folder;<dm_folder>`	
`dm_document`	`Documentum_Document;<dm_document>`	

a. Attribute names are those listed in the "Types" node of the Documentum WebTop interface.

Glossary

This glossary explains the terms used throughout this guide that are specific to Content Integration Platform.

Archiving

The process of storing WebCenter Sites assets to Documentum. Stored assets can be assigned compliance and retention policies. See also cip commander publish command.

Assets

WebCenter Sites assets, both basic and flex.

`cip commander publish` command

CIP command used to initialize the source and target workspaces. The `publish` command replicates the source system's specified workspace to the target system and initializes the synchronization engine, which then starts monitoring the source workspace. The `publish` command is used in both models, publishing and archiving.

CM

Content management, such as creating and editing assets in WebCenter Sites.

CS DataStore

CIP name for the hierarchical folder structure that stores WebCenter Sites assets for archiving on Documentum. The CS DataStore is created and exported during a RealTime publishing session that is enabled for archiving.

Items

General term for WebCenter Sites assets and Documentum objects.

mappings.xml

The metadata map that enables Content Integration Platform to interpret Documentum objects as WebCenter Sites assets and vice versa.

Monitored Workspace

A published or archived workspace. The workspace is monitored by the CIP synchronization engine, which automatically replicates updated content based on previously replicated metadata (defined in mappings.xml).

Objects

Documentum objects, such as cabinets, folders, and documents.

Publishing

Replicating Documentum objects to WebCenter Sites. Publishing does not produce a website. See also cip commander publish command.

Synchronization

The process of keeping content on source and target systems in agreement. The synchronization interval can be configured. *See also* cip commander publish command.

Workspace

An object that stores data to be published or archived. In the publishing model, the source workspace is a Documentum cabinet containing the folder that will be published; the target workspace is a WebCenter Sites flex family (`Documentum` by default). In the archiving model, the source workspace is a CS DataStore; the target workspace is a Documentum folder.