Oracle® WebCenter Sites

Administrator's Guide for Content Integration Platform for File Systems and Microsoft SharePoint

11*g* Release 1 (11.1.1)

February 2012

ORACLE®

Oracle® WebCenter Sites Administrator's Guide for Content Integration Platform for File Systems and Microsoft SharePoint, 11*g* Release 1 (11.1.1)

Primary Author: Tatiana Kolubayev

Contributor: Valentin Vakar, Chandrashekar Avadhani, Suman Saha

Table of

Contents

Part 2. Customizing CIP Installations

Part 3. System Specifications

About This Guide

This guide describes the process of installing Oracle WebCenter Sites: Content Integration Platform for File Systems and Microsoft SharePoint. This guide also describes using the platform to publish to the Oracle WebCenter Sites web application from file systems and Microsoft SharePoint.

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 file systems and Microsoft SharePoint.

Content Integration Platform for File Systems and Microsoft SharePoint 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 the Content Integration Platform for File Systems and Microsoft SharePoint. 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 source systems, their data models, and the WebCenter Sites flex asset model.

Related Documents

For more information, see the following documents:

- *Oracle WebCenter Sites Administrator's Guide*

- *Oracle WebCenter Sites Developer'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.

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

Integrating with Oracle WebCenter Sites

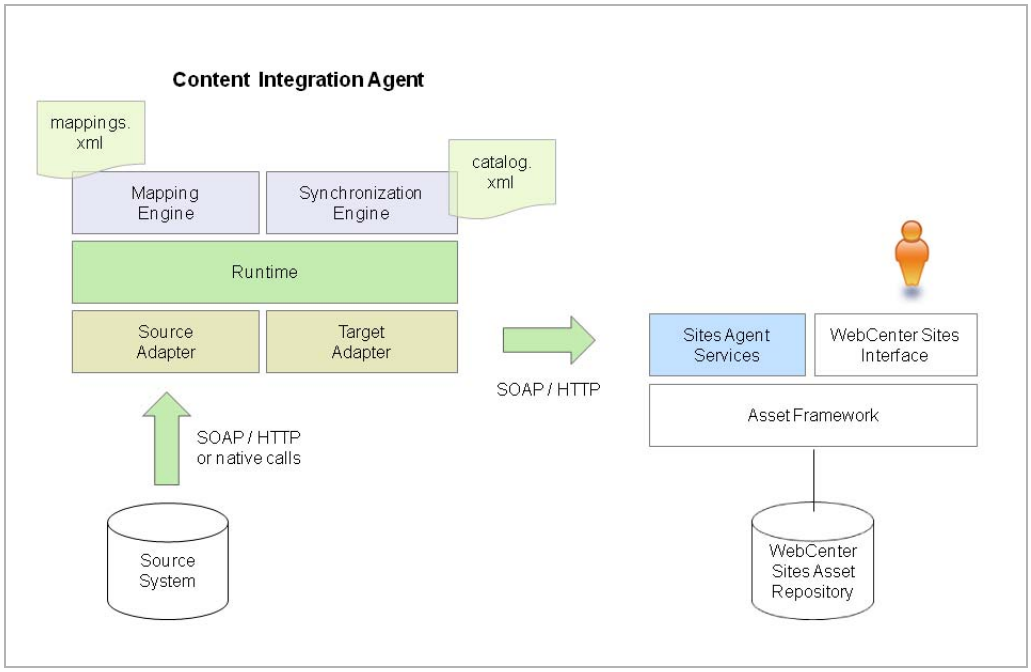
This chapter begins with an overview of the Oracle WebCenter Sites: Content Integration Platform (CIP) for File Systems and Microsoft SharePoint. The middle section provides information about the default mapping framework, which supports default publishing scenarios. The last section outlines your option to set up workflows that inform administrators of events that occur on the target system when monitored content is modified on the source system.

This chapter contains the following sections:

Overview

Content Integration Platform (CIP) for File Systems and Microsoft SharePoint enables you to publish files and SharePoint objects to WebCenter Sites. The publishing process uses the CIP components Content Integration Agent and Sites Agent Services, both shown in Figure 1.

Figure 1: System Architecture



- **Content Integration Agent** is a process, either daemon or standalone, responsible for synchronizing the metadata of objects selected for publication. Its main components are:
  - The `mappings.xml` file, which defines the metadata of objects selected for publication.
  - `CIPCommander`, used to initiate publishing sessions from the command line, where you name the folder to be published. Publishable content includes subfolders and documents (and pictures in SharePoint systems).

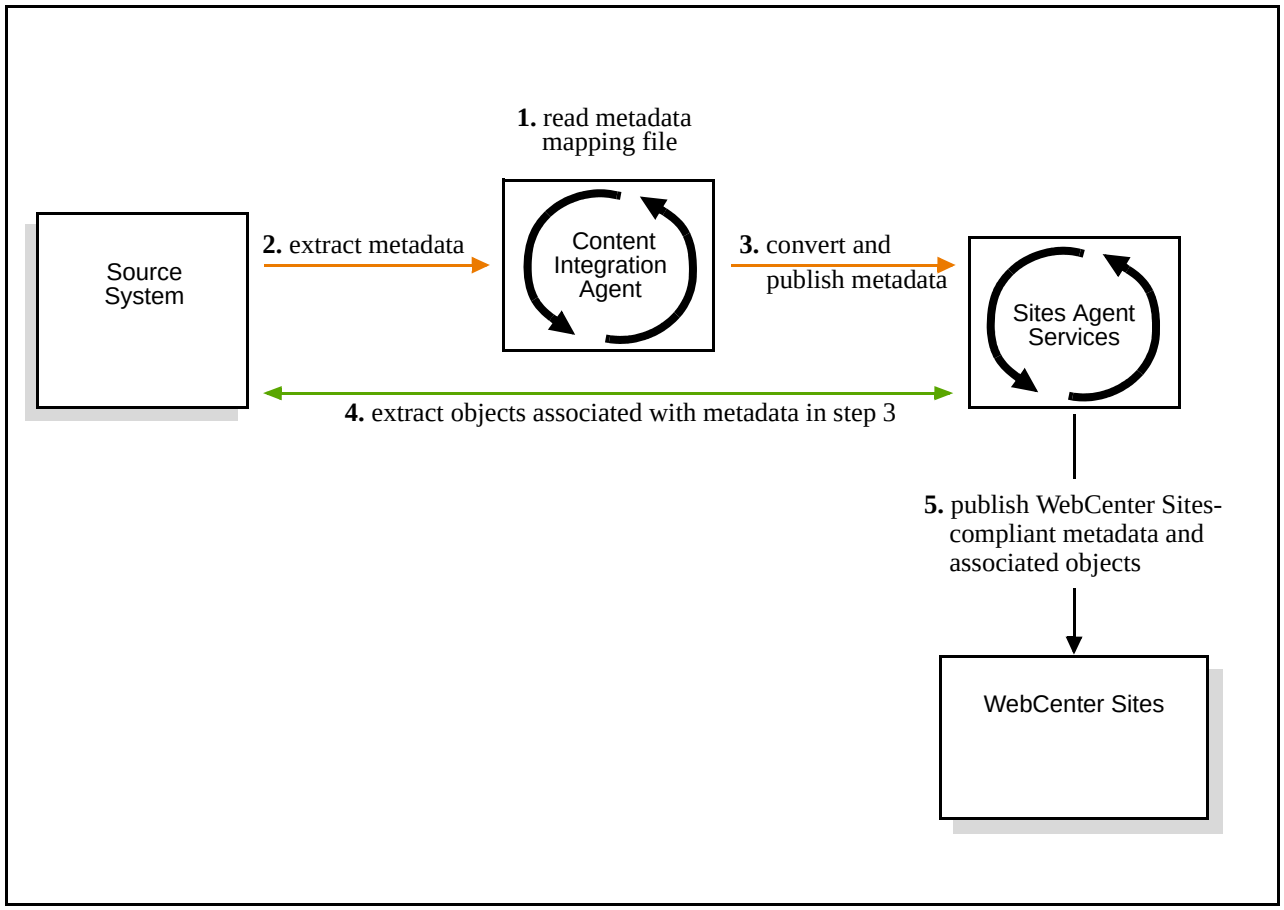
  During a publishing session, the Content Integration Agent process 1) reads the `mappings.xml` file, 2) extracts the mapped metadata from the source system, 3) converts the metadata to a format recognized by WebCenter Sites and invokes the synchronization engine to publish the WebCenter Sites-compliant metadata to Agent Services. Having completed the publishing process, the synchronization engine starts monitoring the source system's published folder. Then, every time a new object is created, deleted, or modified in the monitored folder, the synchronization engine updates Sites Agent Services with the new metadata. (The synchronization interval is configurable.)
- **Sites Agent Services** is a web application responsible for receiving WebCenter Sites-compliant metadata from Content Integration Agent and storing it in the WebCenter Sites database. Agent Services also runs a background process that

extracts and stores the primary binary contents associated with the metadata. Agent Services exposes the Web Services interface needed by Content Integration Agent to perform the synchronization process.

- `catalog.xml` (also within Content Integration Agent), stores information about published objects. When the objects are "unpublished," their information is deleted from `catalog.xml`.

The publishing process, illustrated in Figure 2, also applies to the synchronization event that takes place when published objects are modified or deleted, or new objects are added to the monitored repository.

Figure 2: Publishing to WebCenter Sites



System Defaults

Publishing an object to WebCenter Sites requires:

- A flex family to store the source system's published metadata (object types and attributes) and the associated objects.
- A mappings file (`mappings.xml`), which maps the object's metadata to asset types and assets in the flex family.

Default Mapping Framework

The default mapping framework in Content Integration Platform defines publishable objects to be of the type listed in Table 1, with the attributes shown in Table 1.

Table 1: Source-System Metadata Supported by Default

Source System	Default Object Types and Attributes		See Also ...

File System	**File Type**:
	Folder type named `Folder`
	Document type named `Document`
		Attributes: [a]
		`contentAttr`
		`contentURL`
		`FileSize`
			Appendix A, "File Systems: Default Mapping Specifications" for complete specifications

SharePoint	**Content Type**:
	Folder type named `Folder`
	Document type named `Document`
	Picture type named `Picture`
		Attributes: [a]
		Documents:
		`contentAttr`
		`contentURL`
		`File Size`
		Pictures:
		`Date Picture Taken`
		`Description`
		`Keywords`
		`Picture Height`
		`Picture Width`
			Appendix B, "Microsoft SharePoint Systems: Default Mapping Specifications" for complete specifications

a. Attribute names are display names. The following system-defined attributes – `DateCreated`, `DateModified`, and `MimeType` – are less commonly used and therefore omitted from the default `mappings.xml` file and flex family.

To give you a quick start, the default mapping framework provides the following components:

- A flex family named after its source system: `FileSystem` or `SharePoint`. The purpose of each flex family is to store the source system's default metadata (in Table 1) on WebCenter Sites and thus provide asset type tables to store the associated objects as assets.
- A pre-configured `mappings.xml` file, which maps the default object types and attributes in Table 1 to WebCenter Sites assets in the `FileSystem` and/or `SharePoint` flex families. (The `mappings.xml` file is located on the server that hosts Content Integration Agent.)

For complete specifications on the default mapping framework, see the appendices at the end of this guide.

Implications for Publishing and Synchronization

The success of publishing and synchronization processes depends on the `mappings.xml` file and the source system's flex family.

Default mappings.xml and Flex Families

If your source objects are of the types specified in the default `mappings.xml` file (see also Table 1, on page 10), you can publish those objects to WebCenter Sites without having to modify either `mappings.xml` or the default flex family. During publishing, the objects are automatically re-created in their respective flex family as either *flex parent assets* (if they are folders) and *flex child assets* (if they are documents or pictures).

Following the publication process, changes to monitored folders and their contents (on the source side) are propagated to the flex family by the synchronization engine. For example, if you modify or delete published objects on the source side, or you create new objects (of the mapped types) in the monitored folders, your changes will be automatically propagated to the flex family by the synchronization engine.

Custom mappings.xml and Flex Families

If the schema of the source system is modified, the source system's flex family must be updated in order for publishing and synchronization to produce the expected results. The `mappings.xml` file, however, may or may not require updates, depending on the nature of the schema changes. For example:

- If new document types are created for documents that will be published, both the flex family and `mappings.xml` must be updated with the new document types.
- If a newly added attribute will be propagated to WebCenter Sites, that attribute must be added to the relevant flex family and assigned to the relevant asset type definition.

  Mapping an attribute, however, is conditional:

  > **An attribute must be mapped (in `mappings.xml`) if it will be propagated to WebCenter Sites *and* it is named differently on the source and WebCenter Sites systems**. (If attributes are named identically, they can be omitted from `mappings.xml`.)

  Incorrect mapping of attributes does not stop the publication process, but it does produce a warning message and an entry in the log file. The publication process continues by skipping to the next publishable object.

Event Notification Option

When events such as asset creation occur in published folders on the source system, Content Integration Platform responds by synchronizing the target system to the source system. Content Integration Platform can be configured for event notification. Events would then trigger notices to CIP administrators, informing them of the events and whether synchronization took place. Notices are delivered to CIP administrators in a simple workflow process. Content Integration Platform ships with several sample workflows. You have the option to enable any or all of the workflows during the CIP installation procedure.

Part 1

Installing and Publishing

This part contains the following chapters:

- Chapter 2, “Installing Oracle WebCenter Sites: Content Integration Platform”
- Chapter 3, “Publishing”

Chapter 2

Installing Oracle WebCenter Sites: Content Integration Platform

This chapter contains procedures for installing and configuring the Content Integration Platform to support publishing from file systems and Microsoft SharePoint systems.

This chapter contains the following sections:

Installation Overview

Prerequisites

- Microsoft Visual C++ 2008 redistributable (x86), which can be downloaded from `http://www.microsoft.com`
- OpenSSL, which can be downloaded from `http://www.openssl.org`

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)	These files install Content Integration Agent (`CIPCommander` and service), and configuration files used to control the Agent process.
`csagentservices.war`	This file installs Sites Agent Services, including property files used to set the detail of log files and regulate access to the WebCenter Sites database.
`cs_filesystem_schema.zip`	This file installs the `FileSystem` flex family.
`cs_sharepoint_schema.zip`	This file installs the `SharePoint` flex family.

Where to Install the Files

Content Integration Platform uses native executables.

To install Content Integration Platform

1. Install Content Integration Agent on any computer that runs a supported operating system and can access both the source and target systems.

   Note

   For information about supported systems in this release, see the *Oracle WebCenter Sites Certification Matrix*.

2. Deploy `csagentservices.war` to a system that has access to the WebCenter Sites `Shared` directory.

   The WebCenter Sites system must be fully functional. **It must not be a production (delivery) system.**

3. Install the applicable schema on WebCenter Sites:
   - `cs_filesystem_schema.zip`
   - `cs_sharepoint_schema.zip`

Complete installation instructions are provided in the next section, "Installing Content Integration Platform."

Installing Content Integration Platform

Note

To ensure a smooth installation process, read the steps below to gain an understanding of the installation procedure and the information you will be asked to provide. To efficiently complete "Step III. Installing Schema on WebCenter Sites," you must be an experienced WebCenter Sites administrator.

In this section, you will complete the following steps:

Step I. Installing Content Integration Agent

1. If you are using a Windows operating system, install Microsoft Visual C++ 2008 Redistributable Package (x86) on the same computer that will host Content Integration Agent. (The redistributable package is available for download from the Microsoft web site.)
2. Run the `cipagent` file on a computer that runs a supported operating system and can access both the source and target systems.
   - **Windows:**

     Run `cipagent-1.5.0.msi` and follow the steps on the screen.

     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-1.0.0.rpm.bin
     ```

This command installs the following directories:

```
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 are required by Content Integration Agent to connect to the source system and WebCenter Sites. You will edit this file to provide Content Integration Agent with system location and user information.

   a. Using a text editor, open `catalog.xml`.

   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 are the name of the host and port where WebCenter Sites is running. Do not alter the context name and context-related path unless you are sure 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` as the username).
      - **context**: Leave this blank

   c. If you are using Microsoft SharePoint, edit the adapter for the SharePoint installation.

      Locate the provider element with name "`sharepoint`" and id "`7137dd5d-9ed7-4327-b4fd-8caeebd5889a`", and set the following parameters:

      - **urlSharepoint:** URL pointing to the SharePoint site from which you plan to publish. Typically you need to modify only the host name (the default value is `http://localhost`).
      - **username**: User name for the account that has permissions to publishable content.
      - **password**: Above user's password.

   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 you have more than one network interface 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. Continue to the next step, "Step II. Installing Sites Agent Services."

Step II. Installing Sites Agent Services

> **Note**
>
> Sites Agent Services can be installed on any WebCenter Sites system other than production (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 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 provided 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, "Step III. Installing Schema on WebCenter Sites."

Step III. Installing Schema on WebCenter Sites

In this step, you will import the applicable zip file(s), listed below, into WebCenter Sites:

- `cs_filesystem_schema.zip`
- `cs_sharepoint_schema.zip`

To install schema

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

Note

To use CatalogMover, you must 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

   c. If necessary, import the remaining files.
3. Log in to the WebCenter Sites Admin interface as the general administrator (**fwadmin / xceladmin**, by default) and continue as follows:

a. Enable each imported flex family for an existing content management site (names of flex family members begin with the name of the source system). You can also create a new site for the flex family (or families).

b. For easy access to published content, create a tree tab (for example, **FileSystem** tab or **SharePoint** tab).

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 IV. Optional. Configuring Event Notification."

Step IV. Optional. Configuring Event Notification

In this step, you will enable any or all of the sample workflows in order to notify CIP administrators of events at the source system and their synchronization.

Note

The following workflows must be installed enabled before an asset is published: `CIPAssetDeleted` and `CIPAssetDeletionFailed`

For information and instructions on installing sample workflows, see Chapter 4, "Configuring Event Notification."

Verifying the Installation

In this step, you will publish a test folder to WebCenter Sites. You will also verify the synchronization process by adding, deleting, and modifying the test folder's contents.

To verify the CIP installation

1. Back up `mappings.xml` (located on the server that hosts Content Integration Agent).
2. Select (or create) a test folder with default metadata (defined in the *default* `mappings.xml` file). That is:
   - The folder type matches the folder type in `mappings.xml` (for quick reference, see Table 1, on page 10).
   - The test folder contains subfolders and documents whose folder type, document type, and attributes match those in `mappings.xml` (or Table 1).

   Note

   If you are verifying the publishing process from a Microsoft SharePoint system, run a test on both the document library and picture library, using `mappings.xml` (or Table 1) to determine the default content types and attributes.

3. Test the publishing process. For instructions, see "Publishing to WebCenter Sites," on page 24.

4. Test the synchronization process by renaming, moving, deleting, and creating subfolders and documents (including pictures in SharePoint). Test the attributes by deleting and adding them. When adding attributes, follow instructions in Chapter 5, "Remapping."
5. Test the unpublish process by running the **`unpublish`** command (page 29).

Publishing Production Data

The quickest way to publish objects is to use the default `mappings.xml` file and flex families provided with CIP. Complete one of the following steps, depending on how your source system is configured:

- Objects are ready for publishing if their schema matches the default schema in `mappings.xml` and the default flex family. Follow the steps in Chapter 3, "Publishing."
- Objects cannot be published successfully if their schema differs from the default schema in `mappings.xml` and the default flex family. Before publishing the objects, remap the schema. This requires you to update the flex family for your source system and reconfigure `mappings.xml`. For instructions, refer to Chapter 5, "Remapping."

Customizing Your Installation

Content Integration Platform can be customized in many ways. For example:

- Default flex families can be modified for custom scenarios, or they can be replaced with custom flex families.
- Flex filters can be added to flex families. (If you implement flex filters, make sure to add the corresponding `jar` files to both the WebCenter Sites and Sites Agent Services applications.)
- Attributes can be modified on the source system, or added to the source system.
- New document types can be added to the source system.
- Workflows can be implemented in order to notify administrators of CIP-related events that occur or fail to occur in WebCenter Sites, in response to changes on the source system.

Information about customizing a CIP installation can be found in Part 2, "Customizing CIP Installations."

Chapter 3

Publishing

This chapter contains information about publishing to WebCenter Sites from file systems and Microsoft SharePoint systems.

This chapter contains the following sections:

Overview

Once the source system schema is mapped to WebCenter Sites, any object that is based on the schema can be published to WebCenter Sites.

Synchronization

Manual publishing is unnecessary after the initial session, because the synchronization engine takes over. The engine monitors published folders (on the source system) and mirrors the changes it detects:

- Modification to published objects
- Deletion of published objects
- Addition of objects to the monitored folder(s), as long as their object types are mapped
- Attributes require special treatment, as explained in Chapter 5, "Remapping."

The synchronization interval can be configured in the **`publish`** command (page 24).

Event notification for the synchronization process can be enabled by installing CIP-related default workflows, or specially creating custom workflows. For more information, see Chapter 4, "Configuring Event Notification."

Unpublishing

The `unpublish` command is used to clear `catalog.xml` of all entries that are associated with published objects. Including the **`-delete`** parameter removes the same entries from the WebCenter Sites database. For more information, see "Unpublish Command," on page 29.

Publishing to WebCenter Sites

If the source system schema is mapped to WebCenter Sites, you can successfully publish objects that are based on the schema. (Otherwise, you will need to remap the schema. Instructions are available in Chapter 5, "Remapping.")

To publish to WebCenter Sites

Note

If you changed the port in step 5 on page 19 (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. Make sure Content Integration Agent is running.
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_repname <source_repname>
  -source_path <source_path>
  -target_repname <target_repname>
  -mapping <mapping_id>
  -replic_mode <full | ingestion>
  -bulk_resynch_interval <seconds>
```

where:

- **`<source_providerid>`** is the provider ID for the source system:

  File System:
  `2023b849-688e-4009-af1f-903fe62d85b7`

  Microsoft SharePoint:
  `7137dd5d-9ed7-4327-b4fd-8caeebd5889a`

- **`<target_providerid>`** is the WebCenter Sites provider ID:

  `70b1e307-26a1-499c-9295-cf0b6bd01342`

Parameters and values for the **`publish`** command are defined in Table 2, "Publishing Parameters."

Examples on the usage of the `publish` command are available on page 27.

Information about an object's publication data is available in "When an Object is Published," on page 28.

Table 2: Publishing Parameters

Publishing Parameter	Value
`-source_repname`	`<source_repname>`: Name of the source repository from which content will be published. Enter the name exactly as it appears in the URL. **Legal values:** **File System:** Empty string (**""**). **SharePoint:** Name of the document library or picture library that contains the folder to be published. **Examples:** See page 27.
`-source_path`	`<source_path>`: Path to the object you want to publish. **Legal values:** **File System:** `/<folder>/<folder>/ ... /<folder>/` (to publish the last folder in the path) **SharePoint:** `/` (to publish a library [document or picture] and its contents) `/<folder>/<folder> ... /<folder>/` (to publish the last folder in the path)
`-target_repname`	`<target_repname>`: Name of the content management site (in WebCenter Sites) in which the flex family is enabled for this source system.

Table 2: Publishing Parameters

Publishing Parameter	Value
`-mapping`	`<mapping_id>`: Value of the `mapping id` in `mappings.xml`. If you are using the default `mappings.xml`, enter one of the following values: • `sharepoint2cs` • `filesystem2cs`
`-replic_mode`	`full` | `ingestion` • `full` means that a full replication will be performed (by default). • `ingestion` means that only item creation events will be propagated. Modifications and deletions on the source side will not be reflected on the target.
`-bulk_resynch_interval`	`<seconds>`: Number of seconds between two successive synchronization events. An optional publishing parameter. For more information, see “Synchronization,” on page 28.

Examples

Publishing from a File System

To publish the `C:\publish` folder to the "CIPDemo" content management site, using `filesystem2cs` default mapping:

```
cipcommander publish 2023b849-688e-4009-af1f-903fe62d85b7
  70b1e307-26a1-499c-9295-cf0b6bd01342
  -source_repname ""
  -source_path c:\publish
  -mapping filesystem2cs
  -target_repname CIPDemo
```

Publishing from Microsoft SharePoint

- To publish the `Images` picture library to the "CIPDemo" content management site using, `sharepoint2cs` default mapping:

```
cipcommander publish 7137dd5d-9ed7-4327-b4fd-8caeebd5889a
    70b1e307-26a1-499c-9295-cf0b6bd01342
  -source_repname Images
  -source_path /
  -mapping sharepoint2cs
  -target_repname CIPDemo
```

- To publish the `Cool/Bright` folder in the `Images` picture library to the "CIPDemo" content management site, using `sharepoint2cs` default mapping:

```
cipcommander publish 7137dd5d-9ed7-4327-b4fd-8caeebd5889a
    70b1e307-26a1-499c-9295-cf0b6bd01342
  -source_repname Images
  -source_path /Cool/Bright
  -mapping sharepoint2cs
  -target_repname CIPDemo
```

When an Object is Published

When an object is published, `catalog.xml` (located in `integration_agent/conf/`) is updated with an entry that identifies both the source system and the WebCenter Sites system (within the `<workspace>` tags), and specifies replication data for the published object (within the `<replication>` tag). The code below is an example of a publication entry for an object that was published from a file system:

```
<workspace id="8a55488d-97c9-4290-92e1-d7bb9e476dc7">
  <provider-ref refid="2023b849-688e-4009-af1f-903fe62d85b7" />
  <init-params>
    <param name="repname" />
    <param name="path">c:/temp/CIP_xml/test1/test3</param>
    <param name="repid" />
  </init-params>
</workspace>
<workspace id="f917ddcb-1cbb-46c5-8840-fb194b693629">
  <provider-ref refid="70b1e307-26a1-499c-9295-cf0b6bd01342" />
  <init-params>
    <param name="repname">CIPTest</param>
    <param name="repid">39e1e988-f1a6-4913-a16d-45a1c5ad9976
     </param>
   </init-params>
</workspace>
<replication>
  <link id="04e4f0f2-8535-492b-8590-691a510884d4">
   <source-ref refid="8a55488d-97c9-4290-92e1-d7bb9e476dc7" />
   <target-ref refid="f917ddcb-1cbb-46c5-8840-fb194b693629" />
   <mapping-ref refid="filesystem2cs" />
   <init-params>
     <param name="BulkResynchInterval">600</param>
     <param name="ReplicMode">full</param>
   </init-params>
 </link>
   </replication>
```

Synchronization

Once objects are published, the synchronization engine monitors the status of the folder(s) from which the objects were published and mirrors the changes it detects. For optimal performance, set the synchronization interval to a value that agrees with the frequency of updates to the source system. To set the synchronization interval, include the `bulk_resynch_interval` parameter in the **`publish`** command (page 25).

Unpublish Command

You can unpublish objects from `catalog.xml` alone (and additionally, from WebCenter Sites) by executing the **`cipcommander unpublish`** command with parameters that suit your requirements.

Note

The `unpublish` command clears `catalog.xml` of all entries that are associated with published objects (for a sample publication entry, see the code on page 28). To remove the same entries from the WebCenter Sites database, you must include the **`-delete`** parameter.

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

```
cipcommander unpublish <parameters>
```

Table 3: Unpublish Parameters

Unpublish Parameter	Description
`-all`	Use this parameter to clear `catalog.xml` of **all** publication entries.
`-linkid`	Use this parameter to clear `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 28). For example: • To unpublish a single object from `catalog.xml`, obtain its <`linkid`> and issue the following command: **`cipcommander unpublish linkid 04e4f0f2-8535-492b-8590-691a510884d4`** • To unpublish multiple objects, add their **`linkid`**'s to the **`unpublish`** command.
`-delete`	Use this parameter to remove, from the WebCenter Sites database, the same objects that you are unpublishing from `catalog.xml`. **Legal values:** `<true | false>` **Default value:** `true`

Part 2

Customizing CIP Installations

This part contains the following chapters:”

- Chapter 4, “Configuring Event Notification”
- Chapter 5, “Remapping”
- Chapter 6, “Adding Assets to a Flex Family”

Chapter 4

Configuring Event Notification

This chapter contains the following sections:

Overview

Event notification keeps Content Integration Platform administrators informed about the synchronicity of source and target systems. When changes are made to monitored folders (published folders on the source system), CIP administrators receive confirmation that the same changes were either successfully propagated to WebCenter Sites, or propagation failed to occur. Event-driven notices are delivered to administrators in a simple workflow process.

A default workflow is available for each of the following events:

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

Custom Workflows

Although CIP-related workflows can be created, in most cases it is more convenient to use the sample workflows provided with Content Integration Platform. If you wish to install sample workflows, continue reading this section. If you wish to create custom workflows, see the *Oracle WebCenter Sites Administrator's Guide* for instructions.

Installing Sample Workflows

Note

If you have already published to WebCenter Sites, install and enable only the following workflows for the published content: `CIPAssetCreated`, `CIPAssetModified`, and `CIPAssetModificationFailed`. The remaining workflows (`CIPAssetDeleted` and `CIPAssetDeletionFailed`) must be installed and enabled before an asset is published.

To install sample workflows

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 sample 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 installed, the following message will be displayed:

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

Verifying Sample Workflows

When the sample workflows are created, associated objects are also created in WebCenter Sites.

To verify the sample workflows and associated objects

1. Log in to the WebCenter Sites Admin interface as an administrator.

2. Verify that the following objects have been created:
   - `CIPAdmin` role, which will be used as the management role for all CIP workflows. All users with the `CIPAdmin` role will be notified about all CIP events in the sample 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 Sample Workflows

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

To enable a CIP workflow

1. Uncomment the following section for each asset type that you wish to enable for event notification:

```
<descriptor-mapping sourceid="assetCreatedProcess"
   targetid="CIPAssetCreated" type="static" />
<descriptor-mapping sourceid="assetModifiedProcess"
   targetid="CIPAssetModified" type="static" />
<descriptor-mapping sourceid="assetDeletedProcess"
   targetid="CIPAssetDeleted" type="static" />
<descriptor-mapping sourceid="assetDeletionFailedProcess"
   targetid="CIPAssetDeletionFailed" type="static" />
```

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

   Sample workflows behave in the following way:

   - For creation, deletion failure, modification, and modification failure events, a task is assigned to all WebCenter Sites users with the `CIPAdmin` role. The task is

simply a way of notifying the users of events and their propagation to the target system. The task can be removed; there is no obligation to take a step.

- When an event occurs, only the first step of the corresponding workflow is taken. If the option “Assign from list of participants” for the first step is chosen, all members of the selected roles will be assigned the next task. (For simplicity, notification workflows comprise one state and two steps.)
- For deletion events, all members with the `CIPAdmin` role receive an email notice. Because the asset no longer exists once the deletion event occurs, no tasks are displayed.

Chapter 5

Remapping

This chapter contains the following sections:

When to Remap

If you plan to publish from a source system whose schema differs from the default schema defined by the Content Integration Platform, you will have to update the flex family corresponding to the source system and possibly `mappings.xml`, depending on how the schema differ. Table 4 lists commonly made schema changes and the pages where you can find procedures for updating the relevant components.

Table 4: Common Schema Changes

Common Schema Changes	For Remapping Procedures, See ...	
	File System	MS SharePoint
Add new attribute	page 41	page 43
Add new document type		
Add new document type and attribute		
Add new picture type		
Add new picture type and attribute		

Note

Procedures in the rest of this chapter call for updates to the default `mappings.xml` file and the default flex families, both supplied as part of the Content Integration Platform.

If you wish to create your own flex family, see the *Oracle WebCenter Sites Developer's Guide* for instructions. When creating your own flex family, keep in mind the source system's default flex family as a model. Basic procedures for remapping to a custom flex family remain the same (except for the names you choose for the family and its members).

If you create flex filters (for either a default flex family or a custom flex family), makes sure to add the corresponding `jar` files to both the WebCenter Sites and the Sites Agent Services applications.

Remapping Procedures

File Systems

If you create new attributes for documents of type `Document` and wish to propagate the attributes to WebCenter Sites, you must update the `FileSystem` flex family with the new attributes. You must also update `mappings.xml` if the new attributes will be named differently on the WebCenter Sites side. (A list of default attributes and document types is available in Appendix A, "File Systems: Default Mapping Specifications.")

To update the flex family and mappings.xml

1. Stop the Content Integration Agent.
2. Follow the remapping procedures in Table 5. For descriptions of tags and parameters, see Table 6, on page 42.
3. Test your changes by running a publishing session. For instructions, see Chapter 3, "Publishing."

Table 5: Remapping File System Schema

1. File System	2. WebCenter Sites	3. Content Integration Agent: mappings.xml
If you add a new attribute.	Create a "FileSystem Attribute" asset. Create the attribute in the `FileSystem` flex family and assign the attribute to the relevant asset type definition. For instructions, see chapter 6 and take note of the "Value" fields in step 5 on page 48.	Map the new attribute in `mappings.xml` **only if the attribute is named differently on the WebCenter Sites side.** Map the attribute within the `<descriptor-mapping>` tag, and nest the tag in the relevant object type mapping (example below).

E.g.,
`CreationDate`
	E.g.,
	`fs_CreationDate`
		E.g.,
		In this example, the `CreationDate` attribute belongs to the `Document` file type. Nest the attribute in the `Document` type mapping (as shown in bold type, below).
		`<assettype-mapping`
		`sourceid="Document"`
		`targetid="FileSystem_Document;fs_document"`
		`id="fs_document">`
		`<descriptor-mapping`
		`sourceid="CreationDate"`
		`targetid="fs_CreationDate"`
		`type="static" />`
		`</assettype-mapping>`

Tags and Parameters for Mapping File System Data to Oracle WebCenter Sites

```
<assettype-mapping
   sourceid="FileType"
   targetid="ChildAssetType | ParentAssetType;
             ChildDefinitionInstance | ParentDefinitionInstance"
   id="assettypeMappingID">
      <descriptor-mapping
          sourceid="FileSystemAttribute"
          targetid="CSAttribute"/>
</assettype-mapping>
```

Table 6: Mapping File System Data to Oracle WebCenter Sites

Tag	Parameter	Parameter Description
`assettype-mapping`	`sourceid`	Specifies the type of document that will be published. **Legal values:** `Document | Folder`
	`targetid`	WebCenter Sites' counterpart to `sourceid`. **For document objects, `targetid` takes the following value:** *`ChildAssetType;ChildDefinitionInstance`* - *`ChildAssetType`*: Stores file system documents. **Default value:** `FileSystem_Document` - *`ChildDefinitionInstance`*: Asset of type `FileSystem Child Definition`. Defines the type of document that will be stored in *`ChildAssetType`*. **Default value:** `fs_document` **Sample value:** `pdfDoc` **Example:** If `sourceid="pdfDoc"` then `targetid` could be the following: `targetid="FileSystem_Document;fs_pdfDoc"` i.e., the (child) asset type named `FileSystem_Document` stores documents of type `fs_pdfDoc`. **Note:** If you were to map a folder type, `targetid` would take the following value: *`ParentAssetType;ParentDefinitionInstance`* - *`ParentAssetType`*: Stores file system folders. **Default value:** `FileSystem_Folder` - *`ParentDefinitionInstance`*: Asset of type `FileSystem Parent Definition`. Defines the type of folder that will be stored in *`ParentAssetType`*. **Default value:** `fs_folder` For more information about the `FileSystem` flex family, see Appendix A, "File Systems: Default Mapping Specifications."
	`id`	Unique value that identifies the asset type mapping. **Default value:** `fs_document | fs_folder`

Table 6: Mapping File System Data to Oracle WebCenter Sites *(continued)*

Tag	Parameter	Parameter Description
`descriptor-mapping`	`sourceid`	Attribute's name in the file system.
	`targetid`	Attribute's name in WebCenter Sites.

Microsoft SharePoint Systems

If you add new attributes and/or document types for the documents you will publish, you must update the `SharePoint` flex family accordingly. You may also have to update `mappings.xml`, as explained in this section. (A list of default attributes and document types is available in Appendix B, "Microsoft SharePoint Systems: Default Mapping Specifications.")

To modify the flex family and mappings.xml

1. Stop the Content Integration Agent.
2. Follow the remapping procedures in Table 7. For descriptions of tags and parameters, see Table 8, on page 45.
3. Test your changes by running a publishing session. For instructions, see Chapter 3, "Publishing."

Table 7: Remapping Microsoft SharePoint Schema

1. MS SharePoint	2. WebCenter Sites	3. Content Integration Agent: mappings.xml
If you add a new attribute.	Create a "SharePoint Attribute" asset. Create the attribute in the `SharePoint` flex family and assign the attribute to the relevant asset type definition. For instructions, see chapter 6 and take note of the "Value" fields in step 5 on page 48.	Map the new attribute in `mappings.xml` **only if the attribute is named differently on the WebCenter Sites side.** Map the attribute within the `<descriptor-mapping>` tag, and nest the tag in the relevant object type mapping (example below).

E.g.,
`CreationDate`
	E.g.,
	`sp_CreationDate`
		E.g.,
		In this example, the `CreationDate` attribute belongs to the `Document` content type. Nest the attribute in the `Document` type definition (as shown in bold type, below).
		`<assettype-mapping`
		`sourceid="Document"`
		`targetid="SharePoint_Document;sp_document"`
		`id="sp_PDF" extends="sp_item">`
		`<descriptor-mapping`
		`sourceid="CreationDate"`
		`targetid="sp_CreationDate"/>`
		`</assettype-mapping>`

Table 7: Remapping Microsoft SharePoint Schema *(continued)*

1. MS SharePoint	2. WebCenter Sites	3. Content Integration Agent: mappings.xml

If you add a new document type.
 	
E.g.,
`PDF`
	Create a "SharePoint Child Definition" asset.
	For instructions on creating child definition assets, see chapter 6.
	
	E.g.,
	`sp_PDF`
		Map the new document type in `mappings.xml`, using the `<assettype-mapping>` tag.
		
		E.g.,
		`<assettype-mapping`
		`sourceid="PDF"`
		`targetid="SharePoint_Document;sp_PDF"`
		`id="sp_PDF" extends="sp_item">`
		`</assettype-mapping>`

If you add a new attribute and new document type.
 	
E.g.,
Attribute:
`CreationDate`
New Document type:
`PDF`
	a. Create a "SharePoint Attribute" asset.
	Create the attribute in the `SharePoint` flex family. For instructions, see chapter 6 and take note of the "Value" fields in step 5 on page 48.
	
	b. Create a "SharePoint Child Definition" asset.
	When creating the child definition asset, assign the new attribute to the asset. For instructions, see chapter 6.
	
	E.g.,
	Attribute:
	`sp_CreationDate`
	New Document type:
	`sp_PDF`
		a. Map the new document type in `mappings.xml`, using the `<assettype-mapping>` tag (example below).
		b. Map the new attribute in `mappings.xml` **only if the attribute is named differently on the WebCenter Sites side.** Map the attribute within the `<descriptor-mapping>` tag, and nest the tag in the relevant object type mapping (example below).
		
		E.g.,
		`<assettype-mapping`
		`sourceid="PDF"`
		`targetid="SharePoint_Document;sp_PDF"`
		`id="sp_PDF" extends="sp_item">`
		`<descriptor-mapping`
		`sourceid=”CreationDate”`
		`targetid=”sp_CreationDate” />`
		`</assettype-mapping>`

If you add a new picture type.	Mapping a new picture type is analogous to mapping a document type. The `targetid` parameter takes the value `SharePoint_Document;<ChildDefinitionInstance>`.	

Tags and Parameters for Mapping Microsoft SharePoint Data to Oracle WebCenter Sites

```
<assettype-mapping
   sourceid="SharePointContentType"
   targetid="ChildAssetType | ParentAssetType;
             ChildDefinitionInstance | ParentDefinitionInstance"
   id="assettypeMappingID" extends="sp_item">
      <descriptor-mapping
          sourceid="SharePointAttribute"
          targetid="CSAttribute"/>
</assettype-mapping>
```

Table 8: Mapping Microsoft SharePoint Data to Oracle WebCenter Sites

Tag	Parameter	Parameter Description
`assettype-mapping`	`sourceid`	Specifies the type of content that will be published. **Default value:** `Document | Picture | Folder`
	`targetid`	WebCenter Sites' counterpart to `sourceid`. **For document and picture objects, `targetid` takes the following value:** *`ChildAssetType;ChildDefinitionInstance`* - *`ChildAssetType`*: Stores SharePoint documents and pictures. **Default value:** `SharePoint_Document` - *`ChildDefinitionInstance`*: Asset of type `SharePoint Child Definition`. Defines the type of document (or picture) that will be stored in *`ChildAssetType`*. **Default value:** `sp_document | sp_picture` **Sample value:** `pdfDoc` **Example:** If `sourceid="pdfDoc"`, then `targetid` could be the following: `targetid="SharePoint_Document;sp_pdfDoc"` i.e., the (child) asset type named `SharePoint_Document` stores documents of type `sp_pdfDoc`. **Note:** If you were to map a folder type, `targetid` would take the following value: *`ParentAssetType;ParentDefinitionInstance`* - *`ParentAssetType`*: Stores SharePoint folders. **Default value:** `SharePoint_Folder` - *`ParentDefinitionInstance`*: Asset of type `SharePoint Parent Definition`. Defines the type of folder that will be stored in *`ParentAssetType`*. **Default value:** `sp_folder` For more information about the `SharePoint` flex family, see Appendix B, "Microsoft SharePoint Systems: Default Mapping Specifications."
	`id`	Unique value that identifies the asset type mapping. By default, *`assettypeMappingID`* takes the following values: `sp_document | sp_picture | sp_folder`

Table 8: Mapping Microsoft SharePoint Data to Oracle WebCenter Sites

Tag	Parameter	Parameter Description
`assettype-mapping` *(continued)*	`id` *(continued)*	The `extends` parameter is a pointer to the mapping id for the next higher-level object in the schema hierarchy. Because the Microsoft SharePoint content model requires that content types always have "`item`" defined as one of their top-level objects, the `extends` parameter will always be used, although it is not technically a required field. Including the `extends` parameter allows attributes defined in parent types to be inherited by child types, starting at the `item` level. By default: `extends=sp_item`
`descriptor-mapping`	`sourceid`	Attribute's name in the Microsoft SharePoint system.
	`targetid`	Attribute's name in WebCenter Sites.

Chapter 6

Adding Assets to a Flex Family

This chapter contains procedures that support the remapping steps in Chapter 5, "Remapping."

This chapter contains the following sections:

Adding a New Attribute

If you define new attributes on the source system and associate them with an object that is or will be published, you must create the corresponding attributes in WebCenter Sites. Otherwise, objects associated with the attributes cannot be properly synchronized (or published).

To add a new attribute to WebCenter Sites

1. Log in to the WebCenter Sites Admin interface as an administrator.
2. Select the site in which the flex family for your source system is enabled.
3. Select **New** from the top navigation bar.
4. Select **New *SourceSystem* Attribute** from the list of options.
5. When filling in fields in the "*SourceSystem* Attribute" form, fill in the following fields as explained below:
   - **Value Type:** Select a value type that is as close as possible to the corresponding attribute's data type on the source system. Although you can map any data type to a string, you gain flexibility by mapping to a similar data type. (For example, a date for a given locale cannot be modified for any other locale unless it is mapped to the `date` type.) If you are using Microsoft SharePoint, use Table 9, on page 49 for suggested conversions.
   - **Number of Values:**
     - If you are using Microsoft SharePoint and re-creating a multivalued attribute, select the multivalued option.
     - For file systems, all attributes are single-valued.
6. Save the attribute.
7. Assign the attribute to the relevant flex parent definition or flex child definition.

SharePoint attributes:

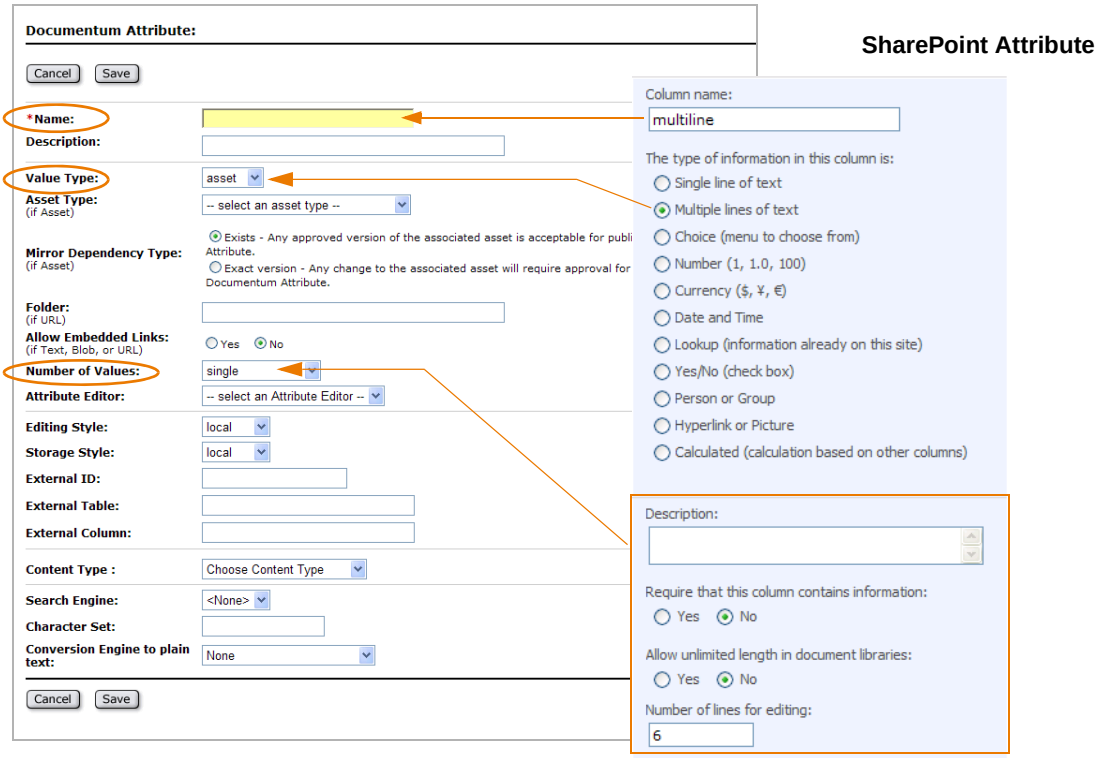


Table 9: Suggested Conversions for SharePoint Attributes

SharePoint Attribute Type	WebCenter Sites Attribute Type	Multi-valued
Single line of text	string / text	
Multiple lines of text	string / text	
Choice (menu to choose from)	string	single- or multi-valued, depending on selection
Number (1, 1.0, 100)	float	
Currency ($, ¥, €)	money	
Date and Time	date	
Lookup (information already on this site)	string	single- or multi-valued, depending on selection
Yes/No (check box)	string	
Person or Group	string	single- or multi-valued, depending on selection
Hyperlink or Picture	string	
Calculated (calculation based on other columns	string	

Adding a New Child Definition Asset

Note

If you are creating a child definition asset with new attributes, create the attributes first (as shown on page 50). You will assign them to the child definition asset during the procedure below.

To add a new child definition asset to WebCenter Sites

1. Log in to the WebCenter Sites Admin interface as an administrator.
2. Select the site in which the flex family is enabled.
3. Select **New** from the top navigation bar.
4. Select **New *SourceSystem* Child Definition** from the list of options and fill in the form that appears.
5. Save the asset.

Part 3

System Specifications

This part contains specifications for the default mappings that come with Oracle WebCenter Sites Content Integration Platform.

This section contains the following appendices:

- Appendix A, “File Systems: Default Mapping Specifications””
- Appendix B, “Microsoft SharePoint Systems: Default Mapping Specifications”

Appendix A

File Systems: Default Mapping Specifications

This appendix contains the following sections:

Overview

The default mapping framework in the Content Integration Platform enables publishing from file systems, as long as the publishable content is based on the following default constructs of the file system: `Folder` and `Document` file types, and attributes `contentAttr`, `contentURL`, and `FileSize`.

The mapping framework supplies the following default components:

- The `FileSystem` flex family, pre-configured to match the file types and attributes listed above.
- A `mappings.xml` file, in which file types and attributes (listed above) are mapped to assets in the `FileSystem` flex family:
  - The `Folder` file type is mapped to a flex *parent* definition asset named `fs_folder`.
  - The `Document` file type is mapped to a flex definition asset named `fs_document`.
  - Attributes are mapped to flex assets of type `FileSystem Attribute`.

  The mapping is illustrated in Figure A-1, on page 55, coded in `mappings.xml`, on page 57, and summarized in Table A-1, on page 58.

Once the mapping is established, folders of type `Folder` (and their contents) can be published:

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

Publishing to the `FileSystem` flex family is summarized below.

During publishing, the Content Integration Platform refers to the `mappings.xml` file to determine the types of objects to publish (`Folder` and `Document`). 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 `FileSystem Folder`, along with all the subfolders and documents it contains.

To reproduce the folder's structure, (subfolders and documents), the Content Integration Platform refers to path information. If subfolders exist, the Content Integration Platform chains their counterpart `FileSystem Folder` assets to reproduce the hierarchy. Documents, treated as flex assets of type `FileSystem Document`, are placed under their respective `FileSystem Folder` parent assets.

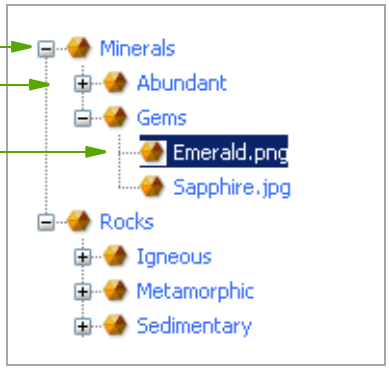
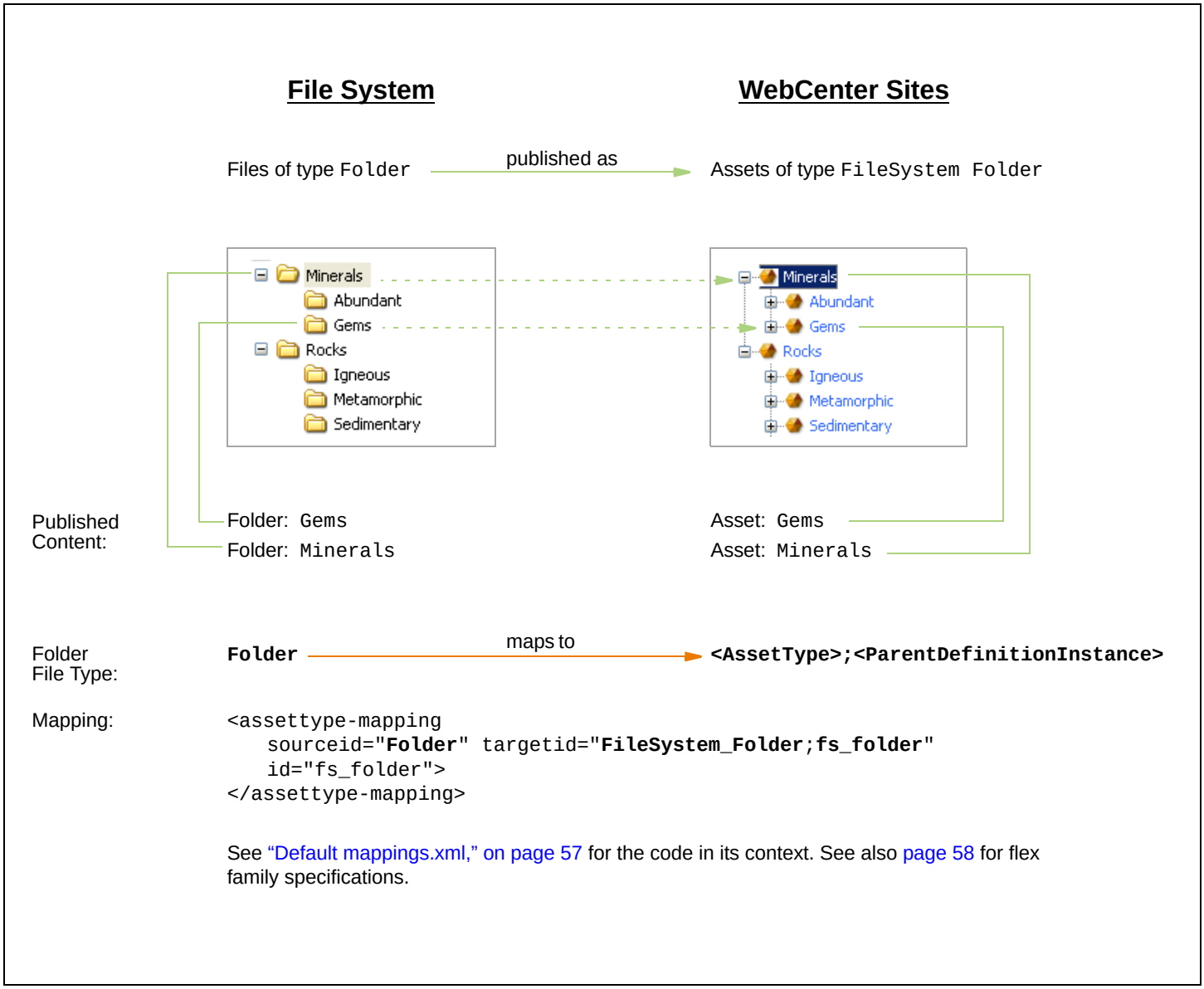
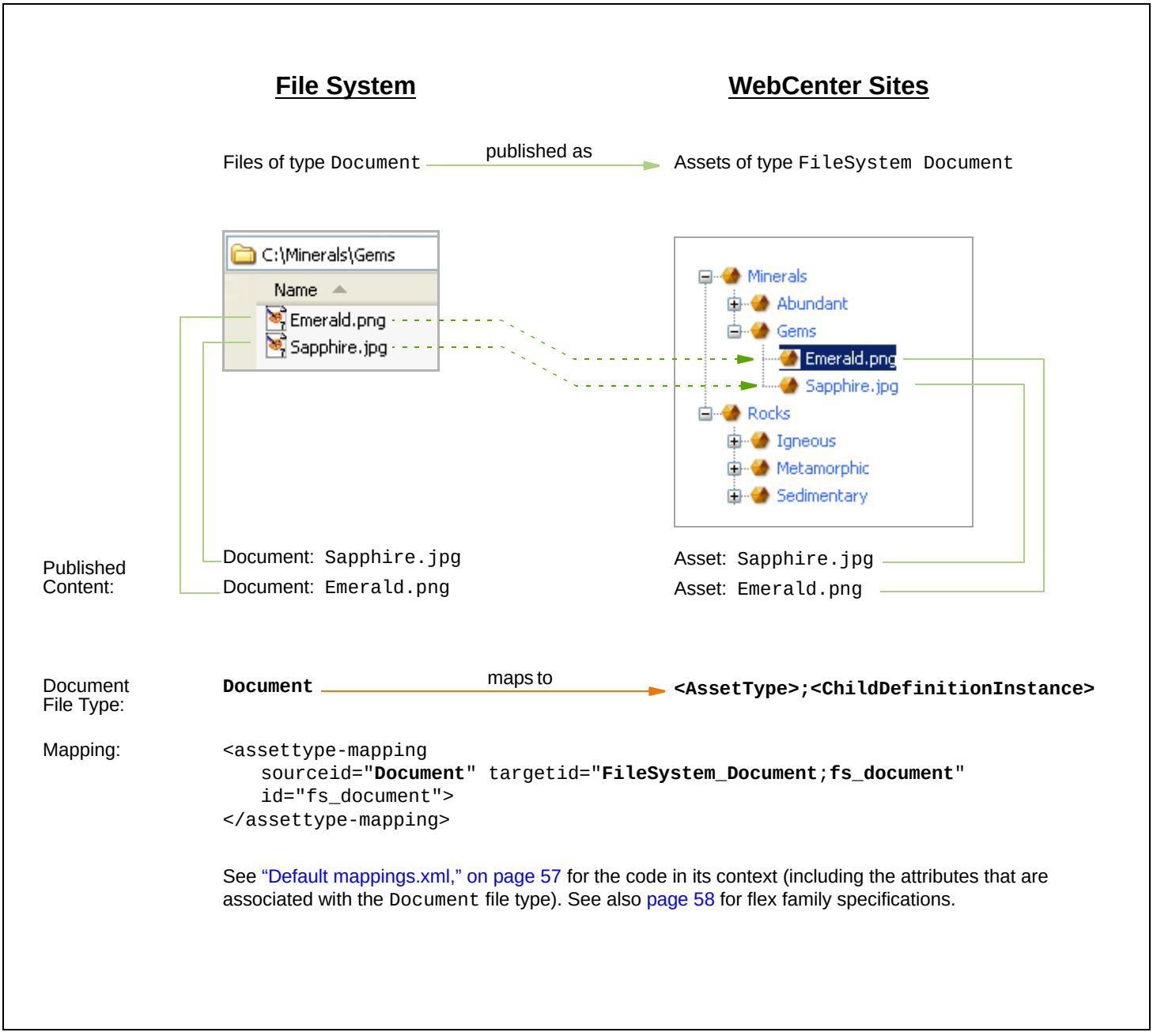


Figure A-1: Mapping the `Folder` Type to the `FileSystem` Flex Family



See "Default mappings.xml," on page 57 for the code in its context. See also page 58 for flex family specifications.

Figure A-2: Mapping the `Document` Type to the `FileSystem` Flex Family



See "Default mappings.xml," on page 57 for the code in its context (including the attributes that are associated with the `Document` file type). See also page 58 for flex family specifications.

Default `mappings.xml`

The default `mappings.xml` file, located in the Content Integration Agent, maps the file system's default schema to WebCenter Sites' `FileSystem` flex family. The mapping is uniquely identified by the ID in line 3. Parameters for the `<assettype-mapping>` and `<descriptor-mapping>` tags are defined in "Tags and Parameters for Mapping File System Data to Oracle WebCenter Sites," on page 42.

```
1  <mappings>
2  <!--  File System to CS   -->
3  <mapping id="filesystem2cs">
4  <assettype-mapping
              sourceid="Document"
              targetid="FileSystem_Document;fs_document"
              id="fs_document">
5    <descriptor-mapping
              sourceid="contentAttr"
              targetid="file" type="static" />
6    <descriptor-mapping
              sourceid="contentURL"
              targetid="contentURL" type="dynamic" />
7    <descriptor-mapping
              sourceid="FileSize"
              targetid="file_size" />
8    </assettype-mapping>
9    <assettype-mapping
              sourceid="Folder"
              targetid="FileSystem_Folder;fs_folder"
              id="fs_folder" />
10   </mapping>
11 </mappings>
```

‘FileSystem’ Flex Family Specifications

Table A-1 summarizes the default mapping of file system schema to WebCenter Sites’ `FileSystem` flex family. For customized implementations, you can either re-use the flex family or create your own.

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

Type of Data	File System Default Data	Maps To:		Description	Assets of This Type Are Created By ...
		Flex Asset Type	Flex Family Member		
Schema	**Attribute** Attributes (for documents):[a] • `contentURL` • `contentAttr` • `FileSize`	**`FileSystem Attribute`** Stores attribute instances:[b] • `contentURL` • `file` • `file_size`	Flex Attribute	This flex asset type stores attributes for documents.	WebCenter Sites Administrator
	Folder File Type `Folder`	**`FileSystem Parent Definition`** Stores parent definition instance: [b] `fs_folder`	Flex Parent Definition	This flex asset type stores parent definition instances.	WebCenter Sites Administrator
	Document File Type `Document`	**`FileSystem Child Definition`** Stores child definition instance: [b] `fs_document`	Flex (Child) Definition	This flex asset type stores child definition instances.	WebCenter Sites Administrator
Content	**Published Folders** Folders of file type `Folder`	**`FileSystem Folder`** Stores flex parent assets (of type `fs_folder` and any other type)	Flex Parent	This flex asset type stores folder assets. For an example, see Figure A-1, on page 55.	Content Integration Platform
	Published Documents Documents of file type `Document`	**`FileSystem Document`** Stores flex (child) assets (of type `fs_document` and any other type)	Flex (Child) Asset	This flex asset type stores document assets. For an example, see Figure A-2, on page 56.	Content Integration Platform

a. Attribute names are display names. The less commonly used system-defined attributes have been omitted from the default `mappings.xml` file and flex family. The attributes are: `DateCreated`, `DateModified`, `MimeType`

b. “Instance” means “asset.” “Instance” is used only to help differentiate metadata (instances) from published content (assets).

Appendix B

Microsoft SharePoint Systems: Default Mapping Specifications

This appendix contains the following sections:

Overview

The default mapping framework in the Content Integration Platform enables publishing from SharePoint libraries on selected sites, as long as the publishable content is based on the following default SharePoint constructs: Content types `Document`, `Folder`, and `Picture`; and attributes `File Size`, `contentAttr`, `contentURL`, `Date Picture Taken`, `Description`, `Keywords`, `Picture Height`, and `Picture Width`.

The mapping framework supplies the following default components:

- The `SharePoint` flex family, pre-configured to match the content types and attributes listed above.
- A `mappings.xml` file, in which content types and attributes (listed above) are mapped to assets in the `SharePoint` flex family:
  - The `Folder` type is mapped to a flex *parent* definition asset named `sp_folder`.
  - The `Document` type is mapped to a flex definition asset named `sp_document`.
  - The `Picture` type is mapped to a flex definition asset named `sp_picture`.
  - Attributes are mapped to flex assets of type `SharePoint Attribute`.

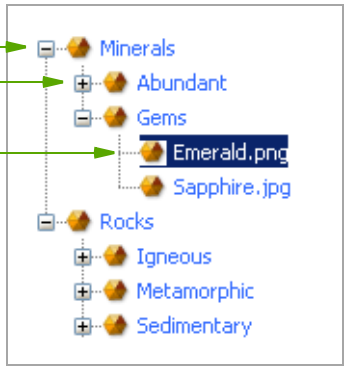
  The mappings are illustrated in Figure B-1, on page 61, coded in `mappings.xml`, on page 63, and summarized in Table B-1, on page 64.

Once the mapping is established, folders of type `Folder` (and their contents) can be published:

- Folders are published as flex *parent* assets to the `SharePoint Folder` asset type.
- Documents and pictures are published as flex assets to the `SharePoint Document` asset type.

Publishing to the SharePoint flex family is summarized below.

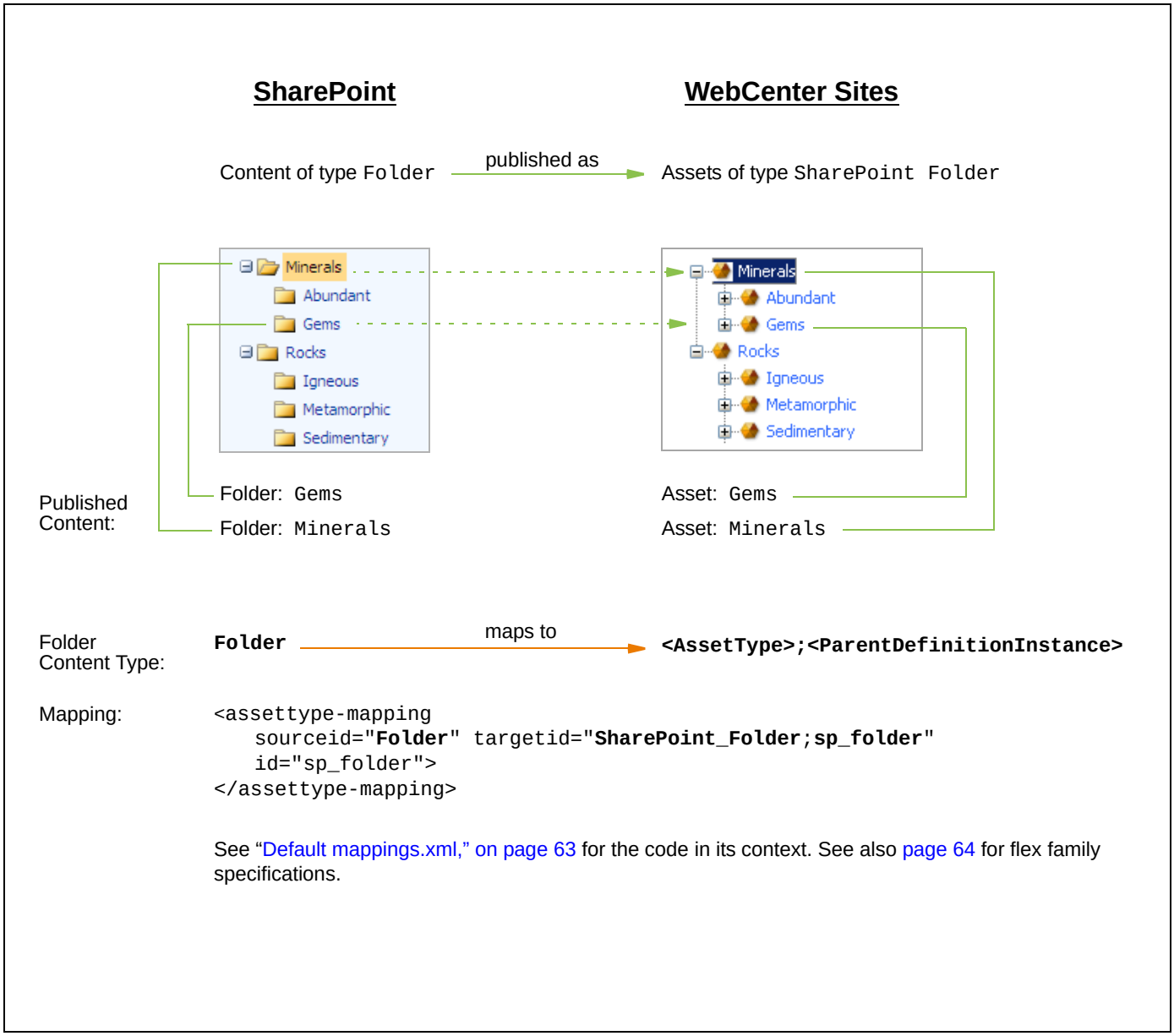
During publishing, the Content Integration Platform refers to the `mappings.xml` file to determine the types of content to publish (`sp_folder`, `sp_document`, and `sp_picture`). The folder that is named in the **`publish`** command is the starting point of the publication process. The folder is published as a flex parent asset of type `SharePoint Folder`, along with all the subfolders, documents, and pictures it contains.



To reproduce the folder's structure, (subfolders, documents, and pictures), the Content Integration Platform refers to path information:

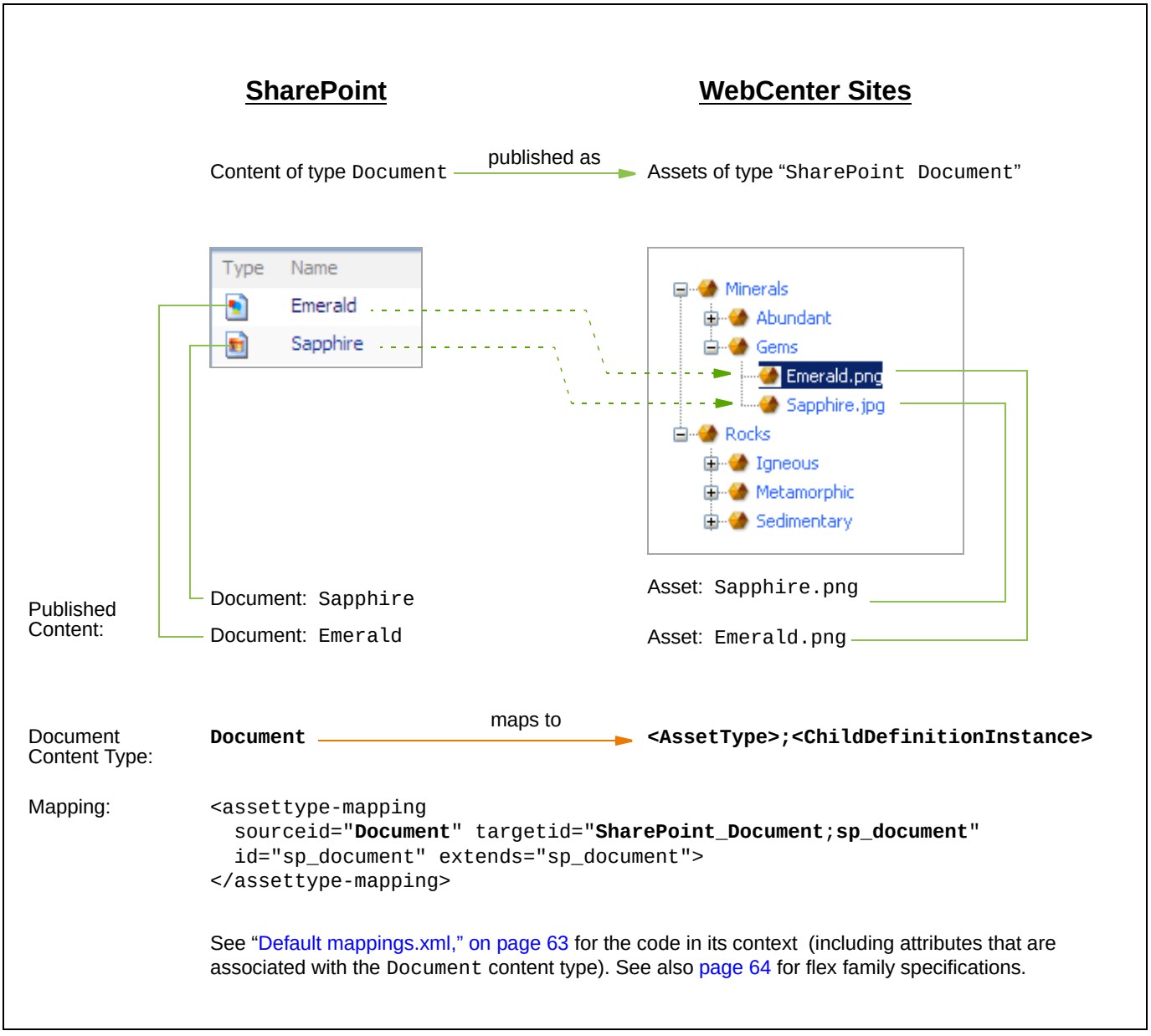
- If subfolders exist, the Content Integration Platform chains their corresponding `SharePoint Folder` assets to reproduce the hierarchy.
- Documents, treated as `SharePoint Document` assets, are placed under their respective `SharePoint Folder` parent assets.
- Pictures, treated as `SharePoint Document` assets, are placed under their respective `SharePoint Folder` parent assets.

Figure B-1: Mapping the `Folder` Type to WebCenter Sites' `SharePoint` Flex Family



See "Default mappings.xml," on page 63 for the code in its context. See also page 64 for flex family specifications.

Figure B-2: Mapping the `Document` Type to WebCenter Sites' `SharePoint` Flex Family



See "Default mappings.xml," on page 63 for the code in its context (including attributes that are associated with the `Document` content type). See also page 64 for flex family specifications.

Note

The `Picture` content type is mapped similarly to the `Document` content type:

- `sourceid` takes the value `Picture`
- `targetid` takes the value `SharePoint_Document;sp_picture`

See "Default mappings.xml," on page 63 for the code (line 13).

Default `mappings.xml`

The default `mappings.xml` file, located in the Content Integration Agent, maps SharePoint schema to WebCenter Sites' `SharePoint` flex family. The mapping is uniquely identified by the ID in line 3. Parameters for the `<assettype-mapping>` and `<descriptor-mapping>` tags are defined in "Tags and Parameters for Mapping Microsoft SharePoint Data to Oracle WebCenter Sites," on page 45.

```
1  <mappings>
2   <!--  MS SharePoint to CS -->
3   <mapping id="sharepoint2cs">
4    <assettype-mapping sourceid="Item" targetid="" id="sp_item">
5     <descriptor-mapping sourceid="Title" targetid="title" />
6    </assettype-mapping>
7    <assettype-mapping
               sourceid="Document"
               targetid="SharePoint_Document;sp_document"
               id="sp_document" extends="sp_item">
8      <descriptor-mapping
               sourceid="File Size"
               targetid="file_size" />
9      <descriptor-mapping
               sourceid="contentAttr"
               targetid="file" type="static" />
10     <descriptor-mapping
               sourceid="contentURL"
               targetid="contentURL" type="dynamic" />
11   </assettype-mapping>
12   <assettype-mapping
               sourceid="Folder"
               targetid="SharePoint_Folder;sp_folder"
               id="sp_folder" extends="sp_item" />
13   <assettype-mapping
               sourceid="Picture"
               targetid="SharePoint_Document;sp_picture"
               id="sp_picture" extends="sp_document">
14     <descriptor-mapping
               sourceid="Date Picture Taken"
               targetid="date_taken" />
15     <descriptor-mapping
               sourceid="Description"
               targetid="description" />
16     <descriptor-mapping
               sourceid="Keywords"
               targetid="keywords" />
17     <descriptor-mapping
               sourceid="Picture Height"
               targetid="picture_height" />
18     <descriptor-mapping
               sourceid="Picture Width"
               targetid="picture_width" />
19   </assettype-mapping>
20  </mapping>
21 </mappings>
```

‘SharePoint’ Flex Family Specifications

Table B-1 summarizes the default mapping of SharePoint schema to WebCenter Sites’ `SharePoint` flex family. For customized implementations, you can either re-use the flex family or create your own.

Table B-1: SharePoint Default Data and Flex Family Analogs

Type of Data	SharePoint Default Data	Maps To: Flex Asset Type	Maps To: Flex Family Member	Description	Assets of This Type Are Created By ...
Schema	**Attribute** Attributes: For documents: • `File Size` • `contentAttr` • `contentURL` For pictures: • `Date Picture Taken` • `Description` • `Keywords` • `Picture Height` • `Picture Width`	**`SharePoint Attribute`** Stores attribute instances: [a] For documents: • `file_size` • `file` • `contentURL` For pictures: • `date_taken` • `description` • `keywords` • `picture_height` • `picture_width`	Flex Attribute	This flex asset type stores attributes for documents and pictures.	WebCenter Sites Administrator
	Folder Content Type `Folder`	**`SharePoint Parent Definition`** Stores parent definition instance: [a] `sp_folder`	Flex Parent Definition	This flex asset type stores parent definition instances.	WebCenter Sites Administrator
	Document Content Type `Document`	**`SharePoint Child Definition`** Stores child definition instance: [a] `sp_document`	Flex (Child) Definition	This flex asset type stores child definition instances.	WebCenter Sites Administrator
	Picture Content Type `Picture`	**`SharePoint Child Definition`** Stores child definition instance: [a] `sp_picture`			
Content	**Published Folders** Folders of content type `Folder`	**`SharePoint Folder`** Stores flex parent assets (of type `sp_folder` and any other type)	Flex Parent	This flex asset type stores folder assets. For an example, see Figure B-1, on page 61.	Content Integration Platform
	Published Documents Documents of content type `Document`	**`SharePoint Document`** Stores flex (child) assets (of type `sp_document` and `sp_picture`, and any other type)	Flex (Child) Asset	This flex asset type stores document assets and picture assets. For an example, see Figure B-2, on page 62.	Content Integration Platform
	Published Pictures Pictures of content type `Picture`				

a. “Instance” means “asset.” “Instance” is used only to help differentiate metadata (instances) from published content (assets).