Content Sync Services Reference Guide

Schneider Electric

Legal Information

Table of Contents

Revision History

Change History

Date	Author	Version	Description
10/22/2013	Todd Burdin	A.1	First release.
11/25/2013	Todd Burdin	A.2	Updated with SharePoint repository and library task. Updated with Query Filter.
11/07/2014	Todd Burdin	A.3	Added HARP support.
7/27/2017	Chris Nitchie	A.4	Added Titania Delivery support; marked original HARP configurations as deprecated.
1/29/2021	Adam Elkins	A.5	

Overview

The purpose of this reference guide is to help an IT administrator install and configure the Content Sync Services. The way this application works is it exports content from Windchill to the local filesystem, them imports it into the target system. In most cases the target system is Titania Delivery. Content Sync is a Java application that is designed to pull content from one location, and push it to another. When it comes to exporting content from Windchill, Titania uses HTTP requests to Windchill using InfoEngine to get the content. All content is exported to a temporary location. If the mirror folder parameter is specified that folder structure is created in that temporary directory. Content Sync exports files to that location, along with metadata xml metadata files named after each file. A flag can be specified to export metadata in a format that is natively supported by Titania as a HARP-META/metadata.json file. XML content has special processing that also exports referenced content and resolves those references but the DTD's must be specified for that to work. Once all the content has been exported from Windchill, the Titania folder sync utility is executed on that folder. A special event handler was written for this Titania folder sync utility to read the XML metadata files and set that metadata in Titania.

Content Sync was primarily designed and is support on Windows platforms. It is a Java applcation though, so it can run on other systems, but Oberon does not officially support those systems.

Content Sync synchronizes batches of content and is not designed to watch Windchill for changes. Most customers run Content Sync via a Windows scheduled task as a nightly job. However, support for syncing content from a Java application is available. The inteded use case is a Windchill workflow, or check-in event. The following is an example of that code. It retrieves the number of a document from the Windchill document and adds an InfoEngine where clause that will be added to the queries made to Windchill for exporting content.

```
try{
    wt.doc.WTDocument doc = (wt.doc.WTDocument)
primaryBusinessObject;
    String tempWorkDir = java.nio.file.Files.
createTempDirectory("titaniasync-").toString().replace("\
\","/");
    System.out.println(tempWorkDir);
    ProcessBuilder pb = new ProcessBuilder("D:\\ptc\
\Windchill_11.0\\Java\\bin\\java.exe",
 "-Xmx512m","-cp", ".\\classes;.\\lib\\*","com.titania.css.
ContentSyncMain",
 "-resourceFolder", ".",
 "-workingFolder", tempWorkDir ,
 "-taskname", "DemoTDSync",
 "-where","number='"+doc.getNumber()+"'");
    pb.directory(new java.io.File("D:\
\WindchillToTitaniaSync\\TitaniaSync-1.3.2-SNAPSHOT\\bin\
\"));Processp = pb.start();
    java.io.InputStream in = p.getInputStream();
    java.io.InputStream err = p.getErrorStream();byte b[]
=new byte[in.available()];
    in.read(b,0,b.length);System.out.println(new String(b));
byte c[]=new byte[err.available()];
    err.read(c,0,c.length);System.out.println(new String
(c));
    in.close();
    err.close();
    p.waitFor();
}catch( Exception ex ){
    ex.printStackTrace();
}
```

Software Prerequisites

The following are 3rd party software packages are required to be installed prior to installing Content Sync Services

- Java JRE 1.7.0_067 or greater

Installation

This section details the steps on how to install **Content Sync Services** components.

Related Topics

- Installation Distribution
- Installing the WindChill Info*Engine Tasks
- Installing the Task Scheduler
- Testing the Installation

Installation Distribution

Unzip the installation distribution to a folder on the target server, for example d:\css. This will be the folder referenced by the Task Scheduler.

The Content Sync Services distribution contains the following:

- The com.titania.repo directory containing the **Windchill Info*Engine** tasks.
- The lib directory containing libraries that need to be deployed.
- The bin directory containing all necessary files to run content sync.

Related Topics

- Installation (Parent Topic)

Installing the WindChill Info*Engine Tasks

Below are detailed steps to install Windchill Info*Engine tasks:

1. Log into the Windchill server as an administrator or as a user that has rights to the Windchill installation.
2. navigate to the Info*Engine tasks folder. This is typically located within the Windchill Installation folder at windchill\tasks .
3. Locate the com.titania.repo folder within the distribution. Copy the entire folder to the tasks folder.
4. Launch the Windchill Command Shell.
5. In the Windchill installation directory, execute the following command to eliminate the maximum query results:.`xconfmanager -s com.ptc.windchill.search.queryLimit=-1 -t \codebase\wt.properties -p`

This completes the installation of the Windchill Info*Engine tasks.

Related Topics

- Installation (Parent Topic)

Installing the Task Scheduler

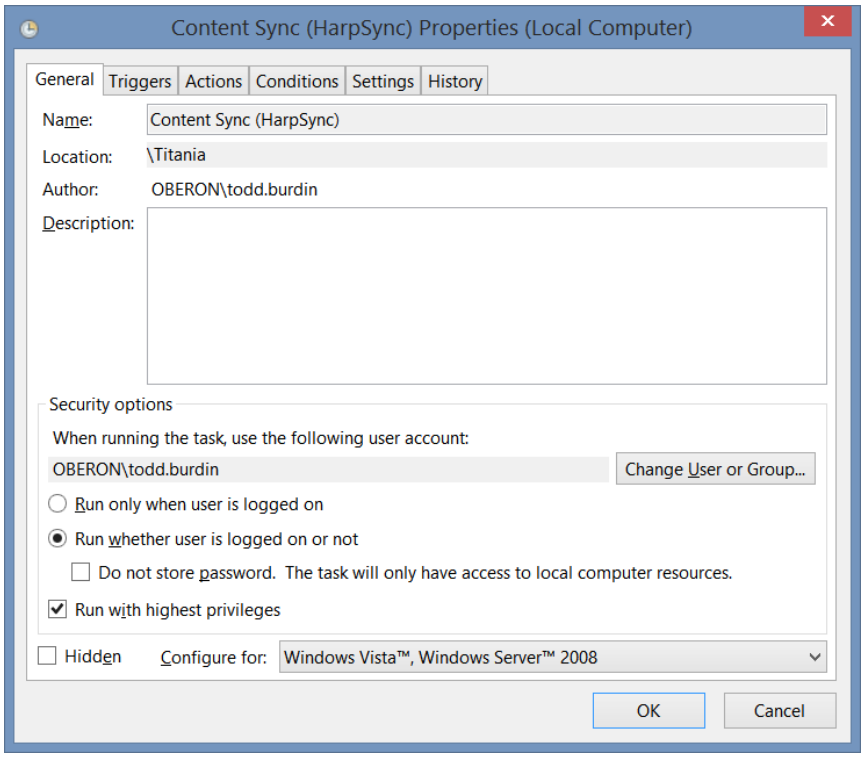
Below are detailed steps to schedule a sync task using Windows Task Scheduler:

1. Locate Task Scheduler script content_sync.bat within the distribution bin .
2. Open the Windows Task Scheduler from within the Administrative Tools.
3. Click on the Task Schedule Library folder in the left pane.

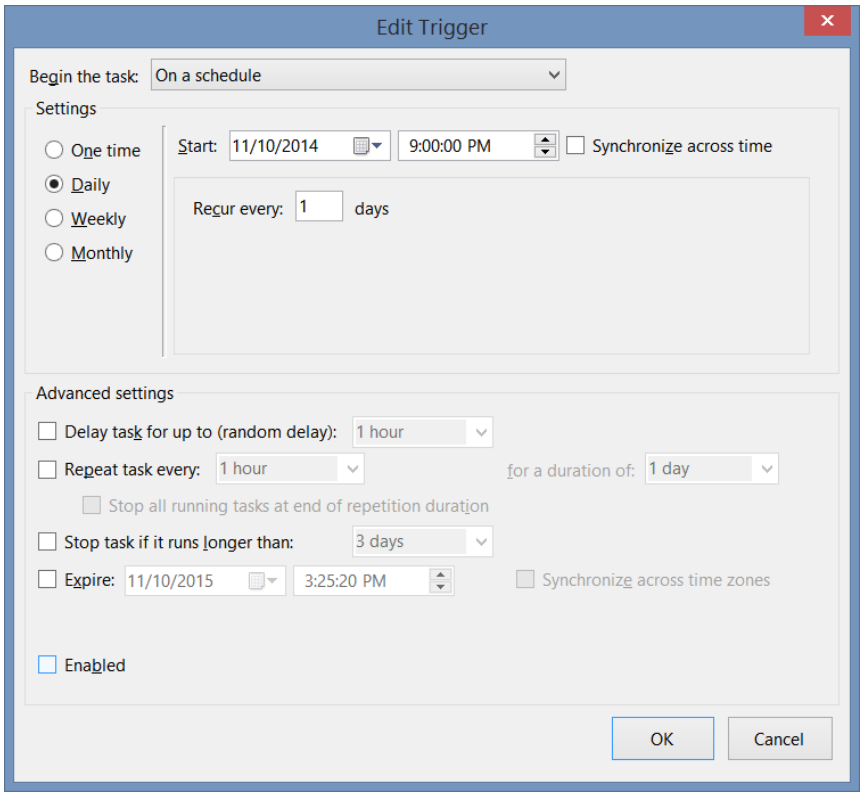
4. From the Action menu, select New Folder. A dialog prompting for name is displayed. Enter a name of a folder that will contain all of the sync tasks. For example, the folder name may be **Titania**. Click the **OK** button to accept. The new folder appears in the left-pane.
5. Click on the new folder and select Create Task from the Action menu. A Create Task dialog appears.

6. Follow the instructions on each tab:

   a. **General** tab: Give the task a name. Select the option to **Run whether user is logged on or not**.

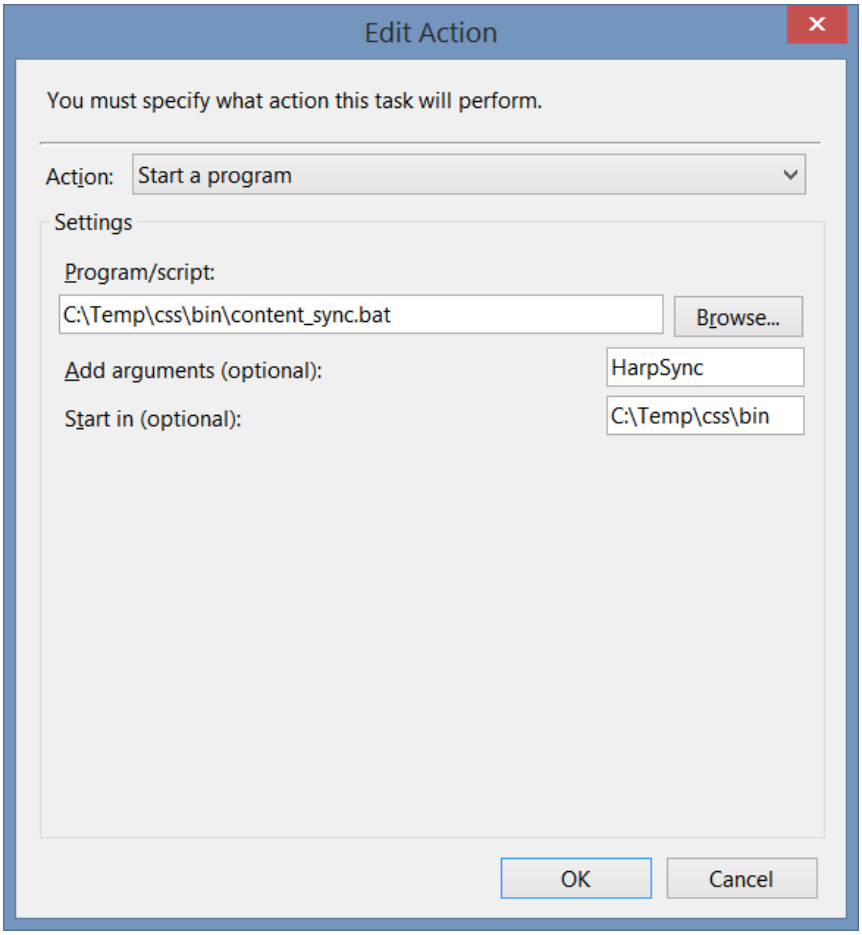


   b. **Triggers** tab: Create a new trigger to run the sync at a specific interval.



   c. **Actions** tab: Provide the name of the sync task to run. The **Program/ script** should be `<distfolder>\bin\content_sync.bat`. Replace <distfolder> with the actual location of the distribution folder.The **Add arguments** prompt should be <taskname>. Replace <taskname> with the sync task name that is defined in the sync-task.xml file.The **Start in** prompt should be

<distfolder>\bin. Replace <distfolder> with the actual location of the distribution folder.



7. Click the OK button to create the new Task

This concludes the installation of the Content Sync Task Scheduler.

Related Topics

- Installation (Parent Topic)

Testing the Installation

Once the system has been configured, it can be tested by manually running the task within Windows Task Scheduler.

Related Topics

- Installation (Parent Topic)

Configuration

Related Topics

- Repository Configurations
- Content Sync Configurations
- Document Types Configurations
- Logging Configuration

Repository Configurations

The Repository Configurations provide the necessary information to enable the Content Sync Services to connect to a source repository. The repository configurations are located in the contentsync web application WEB-APP/repo-config.xml file.

The following sections details the repository configuration options.

Related Topics

- Configuration (Parent Topic)
- Windchill Repository
- SharePoint Repository
- Titania Delivery Repository
- HARP Repository (Deprecated)

Windchill Repository

The Windchill Repository provides a connection to the PTC Windchill content management system. This allows the user to view in real time what is on the Windchill content management system within a local environment.

Attributes

Attribute Name	Description
name	Descriptive name of the repository.
type	“windchill”

Parameters

Parameter Name	Description
url	url of the Windchill instance.
username	Username of the end user that will perform the actions within the repository.
password	Password of enduser that is associated with the username

Example

Below is an example of how to configure a Windchill Repository

```
<Repository name="Windchill" type="windchill">
<Param name='url' value='http://acm.company.com/Windchill'
/>
<Param name='username' value='orgadmin' />
<Param name='password' value='password' />
</Repository>
```

Related Topics

- Repository Configurations (Parent Topic)

SharePoint Repository

The SharePoint Repository provides a connection to the Microsoft SharePoint repository.

Attributes

Attribute Name	Description
name	Descriptive name of the repository.
type	“sharepoint”

Parameters

Parameter Name	Description
url	URL of the SharePoint instance.
username	Username of the end user that will perform the actions within the repository.
password	Password that is associated with the username
domain	Domain that is associated with the username.

Example

Below is an example of how to configure a SharePoint Repository

```
<Repository name="SharePoint" type="sharepoint">
<Param name='url' value='http://sharepoint.acme.com' />
<Param name='username' value='admin' />
<Param name='password' value='password' />
<Param name='domain' value='USCORP' />
</Repository>
```

Related Topics

- Repository Configurations (Parent Topic)

Titania Delivery Repository

The Titania Delivery Repository provides a connection to a Titania Delivery Project.

Attributes

Attribute Name	Description
name	Descriptive name of the repository.
type	“titania-delivery”

Parameters

Parameter Name	Description
url	url of the HARP admin service. For example, *https://qa.harpcontent.com/admin/*
username	Username of the end user that will perform the actions within the repository.
password	Password of enduser that is associated with the username
projectKey	Key for a specific project. This can be obtained by looking at the HARP URL while within the project. For example, *5458dca0efa24ad663b70918*
projectFolder	The folder within the project to push content into. May be blank to push to the root of the project.

requestTimeout	An integer in milliseconds that defines how long the Titania Sync client will wait before timing out when executing HTTP Requests. The maximum number this can be when this was written is 60000 due to that be the max amount of time before the server side issues a timeout.This is useful when syncing to folder in Titania that have thousands of objects.
updateKey	updateKey doesn't do anything on its own, but is used in conjunction with the parameter updateDuplicates, and/or reporcessContexts. It defines the Titania metadata field name that the underlying code will use to find associated objects.
updateDuplicates	updateKey must also be defined. This parameter activates functionality that loops over each object being pushed from Titania to Windchill that queries Titania for other objects with the same updateKey value and updates those objects with the latest version of the object content, and metadata.
reprocessContexts	updateKey must also be defined. This parameter activates functionality loops over each object being synchronized with Titania from Windchill, queries Titania for objects that have the same updateKey in any project, and creates a list of contexts. After all objects have been evaluated, and a unique list of contexts that need to be reprocesssed is created, code iterates over each context and reprocesses them. This code waits up to 5 minutes for each object to be reprocessed and reports the status in the log as it does this.

Example

Below is an example of how to configure a Titania Repository

```
<Repository name="TitaniaQA" type="titania-delivery">
  <Param name='url' value='https://td.example.com/admin/' />
  <Param name='username' value='demo@example.com' />
  <Param name='password' value='Passw0rd' />
  <Param name='projectKey' value='12345' />
  <Param name='projectFolder' value='/qa' />
</Repository>
```

Related Topics

- Repository Configurations (Parent Topic)

HARP Repository (Deprecated)

The HARP Repository provides a connection to a Titania Delivery Project.

> **NOTE:** This repository has been deprecated and may be removed in a future release. Use Titania Delivery Repository, page 13 instead.

Attributes

Attribute Name	Description
name	Descriptive name of the repository.
type	"harp"

Parameters

Parameter Name	Description
url	url of the HARP admin service. For example, *https://qa.harpcontent.com/admin/*
username	Username of the end user that will perform the actions within the repository.
password	Password of enduser that is associated with the username
projectKey	Key for a specific project. This can be obtained by looking at the HARP URL while within the project. For example, *5458dca0efa24ad663b70918*

Example

Below is an example of how to configure a HARP Repository

```
<Repository name="HarpQA" type="harp">
```

```
<Param name='url' value='https://qa.harpcontent.com/admin/'
/>
<Param name='username' value='orgadmin' />
<Param name='password' value='password' />
<Param name='projectKey' value='5458dca0efa24ad663b70918' />
</Repository>
```

Related Topics

- Repository Configurations (Parent Topic)

Content Sync Configurations

The content sync configurations detail one or more tasks used to synchronize content from a specific repository to a destination. The content sync configurations are located in the Tomcat content Sync WEB-INFO\sync-config.xml.

Related Topics

- Configuration (Parent Topic)
- Tasks
- Task Hooks
- Metadata Mapping
- Collections
- Filters

Tasks

Tasks are the primary building blocks of the content sync. All synchronization is defined by a task. There can be one or more tasks that define how content is synchronized. The following sections detail the available types of task configurations.

Related Topics

- Content Sync Configurations (Parent Topic)
- Folder Synchronization Task
- SharePoint Library Sync Task
- Titania Delivery Synchronization Task
- HARP Synchronization Task (Deprecated)

Folder Synchronization Task

The Folder synchronization Task provides the ability to synchronize content located in a repository with the local file system. The following tables detail the Attributes and Parameters along with their description.

Attributes

Attribute Name	Description
name	Descriptive name of the task.
type	“FolderSync”

Parameters

Parameter Name	Description
repository	The name of the repository as defined in the repository configuration. A Parameter is required.
collection	The name of the collection as defined in the collections element. Parameter is required.
fullSync	Indicates if a full or incremental sync should be performed. Valid values are true or false. Default value is “false”.
exportToFolderPath	Location on the local file system where content will be synced to.
mirrorRepositoryFolderPaths	Indicates if the repository folder structure should be mirrored on the local file system. If true, the container and folders will be mirrored. If false, all files will be synced to a single folder.
fileReferenceAttributes	A comma delimited list of all attribute names that contain an xml or graphical reference, delimited by commas. Default value is “href”.
exportAllReferencedGraphics	Indicates if any graphics referenced by the xml is also synced, whether it is in the collection or not.
enforceFileUniqueness	Indicates if the system should ensure that local file names are unique. If true, the Windchill document number will be appended to the filename. If false, only the Windchill filename will be used. Duplicate files will be overwritten. Default value is “true”.
outputMetadataFile	Indicates if the system should generate an XML .properties file along with the exported content file. If true, a properties file having the same base name as the content file will be created and contain all attributes returned by the filter. Additional attributes can be returned by using the filter includeMetadata parameter. If false, no properties file will be created. Default value is “false”.
outputJSONMetadataFile	If set to true, code will be executed to convert metadata to format natively supported by Titania in a HARP-META/metadata.json file.

Contains Elements

•

Example

```
<Task name="Production" type="FolderSync">
<Param name="repository" value="Windchill"/>
<Param name="collection" value="Released Documents"/>
<Param name="fullSync" value="false"/>
<Param name="exportToFolderPath" value="c:\temp\export"/>
<Param name="mirrorRepositoryFolderPaths" value="true"/>
<Param name="fileReferenceAttributes" value="href"/> <!--
separated by commas -->
<Param name="exportAllReferencedGraphics" value="true"/>
<Param name="ensureFileUniqueness" value="true"/>
</Task>
```

Related Topics

- Tasks (Parent Topic)

SharePoint Library Sync Task

The SharePoint Library Synchronization Task provides the ability to synchronize content located in a Windchill repository with a SharePoint document library. The following tables detail the Attributes and Parameters along with their description.

Attributes

Attribute Name	Description
name	Descriptive name of the task.
type	“SPLibrarySync”

Parameters

Parameter Name	Description
repository	The name of the repository as defined in the repository configuration. A Parameter is required.
collection	The name of the collection as defined in the collections element. Parameter is required.
fullSync	Indicates if a full or incremental sync should be performed. Valid values are true or false. Default value is “false”.
mirrorRepositoryFol-derPaths	Indicates if the repository folder structure should be mirrored on the local file system. If true, the container and folders will be mirrored. If false, all files will be synced to a single folder.
spRepository	The name of the SharePoint repository as defined in the repository configuration. A Parameter is required.
site	URL fragment of the site within SharePoint. This will be added to the repository URL to determine the final URL.
libraryTitle	Contains display name of the sync library. The display name is the document library name that is visible to the user.
libraryName	Contains the internal name of the sync document library. The internal name can be found in the URL while viewing the library. The libraryTitle and libraryName may be different.

Contains Elements

-
-

Example

```
<Task name="Production" type="SPLibrarySync">
<Param name="repository" value="Windchill"/>
<Param name="collection" value="Released Documents"/>
<Param name="fullSync" value="false"/>
<Param name="mirrorRepositoryFolderPaths" value="true"/>
<Param name="spRepository" value="SharePoint"/>
<Param name="mirrorRepositoryFolderPaths" value="true"/>
<Param name="site" value="Manuals"/>
<Param name="libraryTitle" value="SME Reviews"/>
<Param name="libraryName" value="Sally Fields"/>
</Task>
```

Related Topics

- Tasks (Parent Topic)

Titania Delivery Synchronization Task

The Titania Delivery synchronization Task provides the ability to synchronize content located in a repository with a Titania Delivery project. It does this by exporting the content from the Windchill repository to the local file system, and then syncing that folder to Titania Delivery. The following tables detail the Attributes and Parameters along with their description.

Attributes

Attribute Name	Description
name	Descriptive name of the task.
type	“TDSync”

Parameters

Parameter Name	Description
repository	The name of the Windchill repository as defined in the repository configuration. Required.
titaniaRepository	The name of the Titania Delivery repository representing the project. Required.

collection	The name of the collection as defined in the collections element. Required.
fullSync	Indicates if a full or incremental sync should be performed. Valid values are true or false. Default value is “false”.
exportToFolderPath	Location on the local file system where content will be synced to. Optional. If not specified, a temporary directory will be created.
cleanup	Indicates whether to delete the folder containing the exported files after the sync is complete. The default is "true" if `exportToFolderPath` is not specified, and "false" if it is.
mirrorRepositoryFolderPaths	Indicates if the repository folder structure should be mirrored on the local file system. If true, the container and folders will be mirrored. If false, all files will be synced to a single folder.
fileReferenceAttributes	A comma delimited list of all attribute names that contain an xml or graphical reference, delimited by commas. Default value is “href”.
exportAllReferencedGraphics	Indicates if any graphics referenced by the xml is also synced, whether it is in the collection or not.
enforceFileUniqueness	Indicates if the system should ensure that local file names are unique. If true, the Windchill document number will be appended to the filename. If false, only the Windchill filename will be used. Duplicate files will be overwritten. Default value is “true”.

Contains Elements

-
-

Example

```
<Task name="DemoTDSync" type="TDSync">
  <Param name="repository" value="DemoWindchillRepo"/>
  <Param name="collection" value="AllDocuments"/>
  <Param name="titaniaRepository" value="DemoTDRepo"/>
  <Param name="mirrorRepositoryFolderPaths" value="true"/>
  <Param name="fileReferenceAttributes" value="href"/>
  <Param name="exportAllReferencedGraphics" value="true"/>
  <Param name="ensureFileUniqueness" value="false"/>
</Task>
```

Related Topics

- Tasks (Parent Topic)

HARP Synchronization Task (Deprecated)

The HARP synchronization Task provides the ability to synchronize content located in a repository with a Titania Delivery project. The following tables detail the Attributes and Parameters along with their description.

> **NOTE:** This task type is deprecated and may be removed in a future release. Use Titania Delivery Synchronization Task, page 17 instead.

Attributes

Attribute Name	Description
name	Descriptive name of the task.
type	“HarpSync”

Parameters

Parameter Name	Description
repository	The name of the repository as defined in the repository configuration. A Parameter is required.
harpRepository	The name of the HARP repository representing the project.
collection	The name of the collection as defined in the collections element. Parameter is required.

fullSync	Indicates if a full or incremental sync should be performed. Valid values are true or false. Default value is “false”.
exportToFolderPath	Location on the local file system where content will be synced to.
mirrorRepositoryFolderPaths	Indicates if the repository folder structure should be mirrored on the local file system. If true, the container and folders will be mirrored. If false, all files will be synced to a single folder.
fileReferenceAttributes	A comma delimited list of all attribute names that contain an xml or graphical reference. Default value is “href”.
exportAllReferencedGraphics	Indicates if any graphics referenced by the xml is also synced, whether it is in the collection or not.
enforceFileUniqueness	Indicates if the system should ensure that local file names are unique. If true, the Windchill document number will be appended to the filename. If false, only the Windchill filename will be used. Duplicate files will be overwritten. Default value is “true”.

Contains Elements

-
-

Example

```
<Task name="Production" type="HarpSync">
<Param name="repository" value="Windchill"/>
<Param name="harpRepository" value="HarpDITAProject"/>
<Param name="collection" value="Released Documents"/>
<Param name="fullSync" value="false"/>
<Param name="exportToFolderPath" value="/Manual"/>
<Param name="mirrorRepositoryFolderPaths" value="true"/>
<Param name="fileReferenceAttributes" value="href"/> <!--
separated by commas -->
<Param name="exportAllReferencedGraphics" value="true"/>
<Param name="ensureFileUniqueness" value="true"/>
</Task>
```

Related Topics

- Tasks (Parent Topic)

Task Hooks

Related Topics

- Content Sync Configurations (Parent Topic)
- Command Line Task Hook
- Java Task Hook

Command Line Task Hook

The command line hook allows for a Windows command file to be executed. The command file is executed in its own thread so must contain all necessary environmental information.

Attributes

Attribute Name	Description
event	Valid values: • **PreTask** – Hook is triggered prior to the task initialization and object collection.. • **PostTask** – Hook is triggered after the task has completed execution.
type	“CommandLineHook”

Parameters

Parameter Name	Description
command	The name of the command file located in the resource folder. If using a relative path, the current folder location is within the web applications WEB-INF folder.

Example:

The following example details how to configure a command line task hook to call a bat file located in the WEB-INF/hooks folder.

```
<Task name="Production" type="FolderSync">
...
<TaskHook event="PostTask" type="CommandLineHook">
<Param name="command" value="hooks/post-task-stub.bat"/>
</TaskHook>
</Task>
```

Related Topics

- Task Hooks (Parent Topic)

Java Task Hook

The Java hook allows custom Java code to be executed.

Attributes

Attribute Name	Description
event	Valid values: • **PreTask** – Hook is triggered prior to the task initialization and object collection.. • **PostTask** – Hook is triggered after the task has completed execution.
type	“CommandLineHook”

Parameters

Parameter Name	Description
class	The fully qualified name of the custom Java class. The custom Java class must implement the `com.titania.css.engine.hook.IJavaHook` interface and be located in the syncs Java classpath.

Example:

The following example details how to configure a Java task hook.

```
<Task name="Production" type="FolderSync">
...
<TaskHook event="PreTask" type="JavaHook">
    <Param name="class" value="com.oberon.acme.css.
SPSyncPreTaskHook"/>
</TaskHook>
</Task>
```

Related Topics

- Task Hooks (Parent Topic)

Metadata Mapping

Metadata mapping enables the sync task to set metadata on the target object based on the source object. This capability only is available if the target object supports file metadata.

Related Topics

- Content Sync Configurations (Parent Topic)
- SharePoint Metadata Mapping
- Titania Delivery Metadata Mapping
- HARP Metadata Mapping (Deprecated)

SharePoint Metadata Mapping

The SharePoint Metadata Mapping allows metadata from a Windchill object to be set into the synced SharePoint field. Metadata can be set from the synced Windchill object itself or a linked object.

Attributes

Attribute Name	Description
type	“WC2SPMapper”

Parameters

Parameter Name	Description
sourceLinkType	The WC link type. For example, the value for a representation link is `wt.representation.PublishedContentLink`. If not specified, metadata from the synced object will be used.
sourceLinkDir	The WC link direction. For example if a representation is being synced, metadata from the authored content can be set by using the value `representable`. If not specified, metadata from the synced object will be used.

Mapping

Attribute Name	Description
source	Windchill attribute name. The attribute must be defined in the filter object type or ancestor object type. Attribute names must be the internal Windchill name.
target	SharePoint field information. The value must be in the format `<FieldDisplayName>~<FieldInternalName>~<FieldType>`.

Example:

The following example details how to configure a metadata mapping to sync content from the authored content.

```
<MetadataMappings type="WC2SPMapper">
<Param name="sourceLinkType" value="wt.representation.
PublishedContentLink"/>
<Param name="sourceLinkDir" value="representable"/>
<Mapping source="iterationInfo.note" target="Comments~_
Comments~Text"/>
<Mapping source="iterationInfo.modifier" target="Component
Author~Component%5Fx0020%5FAuthor~User"/>
</MetadataMappings>
```

Related Topics

- Metadata Mapping (Parent Topic)

Titania Delivery Metadata Mapping

The Titania Delivery Metadata Mapping allows metadata from a Windchill object to be set into the synced Titania Delivery metadata field.

Attributes

Attribute Name	Description
type	“WC2TDMapper”

Parameters

Parameter Name	Description
cascades	Comma-delimited list of metadata names, source or target, that should be marked as cascading in Titania Delivery. Cascading properties on DITA maps apply to the maps as well as contextualized copies of the topics it references.

Mapping

Attribute Name	Description
source	Windchill attribute name. The attribute must be defined in the filter object type or ancestor object type. Attribute names must be the internal Windchill name.
target	Titania Delivery metadata name.

Example:

The following example details how to configure a metadata mapping to sync content from the authored content.

```
<MetadataMappings type="WC2TDMapper">
  <!-- Cascading metadata names, either source or target,
should
       be specified here, comma-delimited. -->
  <Param name="cascades" value=""/>
  <Mapping source="category" target="wc.category"/>
  <Mapping source="createdBy" target="wc.createdBy"/>
  <Mapping source="folder" target="wc.folder"/>
  <Mapping source="iteration" target="wc.iteration"/>
  <Mapping source="modifiedBy" target="wc.modifiedBy"/>
  <Mapping source="revision" target="wc.revision"/>
  <Mapping source="state.state" target="wc.state"/>
  <Mapping source="thePersistInfo.createStamp" target="wc.
created"/>
  <Mapping source="thePersistInfo.modifyStamp" target="wc.
modified"/>
  <Mapping source="thePersistInfo.updateStamp" target="wc.
updated"/>
  <Mapping source="urlLocation" target="wc.url"/>
</MetadataMappings>
```

Related Topics

- Metadata Mapping (Parent Topic)

HARP Metadata Mapping (Deprecated)

The HARP Metadata Mapping allows metadata from a Windchill object to be set into the synced HARP metadata field. Metadata can be set from the synced Windchill object itself or a linked object.

> **NOTE:** This metadata mapping type is deprecated and may be removed in future releases. Use Titania Delivery Metadata Mapping, page 21 instead.

Attributes

Attribute Name	Description
type	“WC2HarpMapper”

Parameters

Parameter Name	Description
sourceLinkType	The WC link type. For example, the value for a representation link is `wt.representation.PublishedContentLink`. If not specified, metadata from the synced object will be used.
sourceLinkDir	The WC link direction. For example if a representation is being synced, metadata from the authored content can be set by using the value `representable`. If not specified, metadata from the synced object will be used.

Mapping

Attribute Name	Description
source	Windchill attribute name. The attribute must be defined in the filter object type or ancestor object type. Attribute names must be the internal Windchill name.
target	HARP metafield name.

Example:

The following example details how to configure a metadata mapping to sync content from the authored content.

```
<MetadataMappings type="WC2HarpMapper">
<Param name="sourceLinkType" value="wt.representation.
PublishedContentLink"/>
<Param name="sourceLinkDir" value="representable"/>
<Mapping source="iterationInfo.note" target="Comments"/>
<Mapping source="iterationInfo.modifier" target="Authors
Name"/>
</MetadataMappings>
```

Related Topics

- Metadata Mapping (Parent Topic)

Collections

Collections define the data set within the repository that will be synced.

Related Topics

- Content Sync Configurations (Parent Topic)
- Basic Collection

Basic Collection

The basic collection provides a wrapper around one or more filters. Within a collection are filters that may select or exclude objects within the repository. Filters act on previously selected objects within the collection. For example, if filter 1 includes objects A and B, then filter 2 excludes object B, then the collection contains only object A. However, if filter 1 excludes object B, then filter 2 includes objects A and B, then the collection contains both object A and B. This is because object B was not yet in the collection to exclude.

Common parameters used by one or more filters can be included as a collection parameter and then inherited by all of the filter parameters. Any collection based parameter may be overwritten by a filter based parameter.

Parameters may also be inherited from the referring task components. The best example is to set the repository in the task and let it be inherited by the filter rather than specifying the repository parameter in each filter.

Attributes

Attribute Name	Description
name	Descriptive name of the collection.
type	“Basic”

Contains Elements

•

Example

The following example defines collection of all released manuals. If a document has a state of CANCELLED in a later revision compared to a RELEASED revision, the document will be removed from the collection.

```
<Collection name="Released Documents" type="Basic">
<Param name="containerName" value="Manuals"/>
<Param name="containerType" value="wt.inf.library.
WTLibrary"/>
<Param name="objectType" value="com.company.
DynamicDocument"/>

<Filter name="IncludeReleased" type="StateFilter">
<Param name="state" value="RELEASED"/>
</Filter>

<Filter name="ExcludeCancelled" type="StateFilter">
<Param name="excludeIfLaterRevision" value="true"/>
<Param name="state" value="CANCELLED"/>
</Filter>
</Collection>
```

Related Topics

- Collections (Parent Topic)

Filters

Filters include or exclude objects from a collection.

Related Topics

- Content Sync Configurations (Parent Topic)
- State Filter
- Query Filter

State Filter

The State Filter queries the repository for objects having a specified state within a revision tree and returns the most recent iteration matching that state.

Below are the attributes and parameters along with a detailed description.

Attributes

Attribute Name	Description
name	Descriptive name of the task.
type	“StateFilter”

Parameters

Parameters are inherited from Collection and referencing elements (i.e. Task).

Parameter Name	Description
containerName	Name of the product or library to be queried.

containerType	Internal object type name of the container to be queried. The out-of-the-box Library container name is “wt.inf.library.WTLibrary”. The out-of-the-box Product container name is “wt.pdmlink.PDMLinkProduct”. If containerName is specified, containerType is required.
state	Internal name of the target state. Required.
objectType	The fully qualified name of the object type to be queried. The default Document object type is “WCTYPE|wt.doc.WTDocument” and the default EPM Document Type is “WCTYPE|wt.epm.EPMDocument”
folder	The container folder to be queried. For example “/Publications”.
criteria	Any additional criteria to be included in the query. The criteria must confirm to the Info*Engine Search-Objects task WHERE Clause. For additional information, refer to the PTC Windchill Adapters Guide.
excludeIfLaterRevision	If the objects returned by the filter are of a later version that what is already been returned by previous filters, if excludeIfLaterRevision is true, the object is removed from the filter. For example, if a document revision history is: B.1 Cancelled A.2 Released A.1 In Work If a previous filter queried for released objects, version A.2 would be in the collection. However, if the a second filter is applied to query for Cancelled objects, returning B.1, and the excludeIfLaterRevision is true, A.2 will be removed from the collection.
includeMetadata	Additional Windchill attributes to be included with the filter results. The attribute must be defined in the object type or ancestor object type. Multiple attributes can be specified delimited with commas. Attribute names must be the internal Windchill name.
syncSubFolders	True or false. Indicates whether to process synchronize subfolders of content. This applies to QueryFilters and StateFilters.

Example

The following example defines collection of all released manuals. If a document has a state of CANCELLED in a later revision compared to a RELEASED revision, the document will be removed from the collection.

```
<Collection name="Released Documents" type="Basic">
<Param name="containerName" value="Manuals"/>
<Param name="containerType" value="wt.inf.library.
WTLibrary"/>
<Param name="objectType" value="com.company.
DynamicDocument"/>

<Filter name="IncludeReleased" type="StateFilter">
<Param name="state" value="RELEASED"/>
</Filter>

<Filter name="ExcludeCancelled" type="StateFilter">
<Param name="excludeIfLaterRevision" value="true"/>
<Param name="state" value="CANCELLED"/>
</Filter>
</Collection>
```

Related Topics

- Filters (Parent Topic)

Query Filter

The Query Filter queries the repository for objects using specified criteria returning the latest objects.

Below are the attributes and parameters along with a detailed description.

Attributes

Attribute Name	Description
name	Descriptive name of the task.
type	“QueryFilter”

Parameters

Parameters are inherited from Collection and referencing elements (i.e. Task).

Parameter Name	Description
containerName	Name of the product or library to be queried.
containerType	Internal object type name of the container to be queried. The out-of-the-box Library container name is “wt.inf.library.WTLibrary”. The out-of-the-box Product container name is “wt.pdmlink.PDMLinkProduct”. If containerName is specified, containerType is required.
objectType	The fully qualified name of the object type to be queried. The default Document object type is “WCTYPE|wt.doc.WTDocument” and the default EPM Document Type is “WCTYPE|wt.epm.EPMDocument”
folder	The container folder to be queried. For example “/Publications”.
criteria	Any additional criteria to be included in the query. The criteria must confirm to the Info*Engine Search-Objects task WHERE Clause. For additional information, refer to the PTC Windchill Adapters Guide.
includeMetadata	Additional Windchill attributes to be included with the filter results. The attribute must be defined in the object type or ancestor object type. Multiple attributes can be specified delimited with commas. Attribute names must be the internal Windchill name.
syncSubFolders	True or false. Indicates whether to process synchronize subfolders of content. This applies to QueryFilters and StateFilters.

Example

The following example defines collection of the latest iteration of two specific manuals.

```
<Collection name="Specific Manuals" type="Basic">
<Param name="containerName" value="Manuals"/>
<Param name="containerType" value="wt.inf.library.
WTLibrary"/>
<Param name="objectType" value="com.company.
DynamicDocument"/>

<Filter name="LatestManual1" type="QueryFilter">
<Param name="criteria" value="number='0000003755'"/>
</Filter>

<Filter name="LatestManual2" type="QueryFilter">
<Param name="criteria" value="number='0000003758'"/>
</Filter>
</Collection>
```

Related Topics

- Filters (Parent Topic)

Document Types Configurations

This section details how to configure a custom document type. The Content Sync Services comes pre-configured for DITA 1.2. Any other document types must be configured.

Document types are defined using XML catalogs and are located within web application WEB-INF/doctypes folder.

Related Topics

- Configuration (Parent Topic)
- Creating a Custom Document Type
- Referencing Arbortext Document Types

Creating a Custom Document Type

- All custom doctypes are located in the WEB-INF/doctypes/custom folder.
- Each document type dependencies should be placed in a single folder within the custom folder.
- A catalog file within each doctype folder should be created that references each DTD/schema.
- A master catalog file should be created in the custom folder that references each doctype catalog file.

Refer to the sample doctype as an example.

Related Topics

- Document Types Configurations (Parent Topic)

Referencing Arbortext Document Types

If all of the DTD/schemas are available in an Arbortext application or custom folder, references to these doctypes can be made from the content sync service catalog file. Arbortext catalogs cannot be referenced directly to resolve URI's as they are not formatted correctly. Following is the procedure to configure content sync services to use the Arbortext document types.

An ACL file named generate-catalog.acl is available in the WEB-INF/tools folder.

1. Locate the ACL script generate-catalog.acl in the content sync web application WEB-INF/tools folder.
2. Open Arbortext Editor, ensuring the custom document type is loaded.
3. Source the generate-catalog.acl from the command line as located in Step 1. A new window will appear containing the generate catalog. The catalog can be used as is or can be modified by removing the unneeded entries.
4. Save the catalog as catalog.xml to the WEB-INF/doctypes/custom folder. The master catalog file is already configured to reference the custom catalog file.

Related Topics

- Document Types Configurations (Parent Topic)

Logging Configuration

There are several different log files utilized by the content sync services. The following lists the log files in the order of importance.

Task Logging

Task-specific log files are located in the web service WEB-INF/working/logs folder. These log files are named based on the task name and a summary of the sync actions. The level of detail in this log file can not be changed.

Content Sync Logging

The content sync log file is named ContentSync.log and is located in the web service WEB-INF/working/logs folder. This log file contains logging for all tasks and the overall content sync service. The level of detail in the log file can be controlled by the WEB-INF/log4j.properties file.

Related Topics

- Configuration (Parent Topic)

Web Services

Content Sync Services provides services that can be utilized by other applications or within repository workflows. All services are built using a REST architecture.

The following sections detail each of the available web services.

Related Topics

- Sync Task
- Object Sync Status

Sync Task

The Sync Task service initiates a syncronization job for a specific task.

Name: SyncTask

Arguments

Argument	Description
taskname	The name of a task as defined in the sync configuration file.

Returns

SUCCESS: Job is complete. Any errors that may have occurred are found in the task log file.

Example

The following URL initiates a sync job for the task named Production.
`http://localhost:8080/contentsync/SyncTask?taskname=Production`

Related Topics

- Web Services (Parent Topic)

Object Sync Status

The Object Sync Status service queries the content sync service for the sync status of a specific object.

Name: ObjectStatus

Arguments

Argument	Description
taskname	The name of a task as defined in the sync configuration file.
obid	The repository object ID of the object being queried.

Returns

COMPLETED: Indicates that the object has been synced.

PENDING: Indicates that the object is still waiting to be synced.

Example

The following URL queries the sync service for the status of an object being sync by the Production task.

```
http://localhost:8080/contentsync/ObjectStatus
?taskname=Production
&obid=VR:wt.epm.EPMDocument: ... acm.company.com
```

Related Topics

- Web Services (Parent Topic)

Printed in:

As standards, specifications, and design change from time to time,
please ask for confirmation of the information given in this publication.