Rosetta-Voyager Synchronization Guide

Version 2.2

The bridge to knowledge

Table of Contents

Chapter 1 Overview of Rosetta-Voyager Synchronization 5

Chapter 2 CMS Integration 7

Introduction 7

Management of the CMS ID 8

Searching and Retrieving Records in CMS Through SRU Protocol 8

Search by Title (Manual Assignment) 8
Search by ID 9
SRU Response 11

CMS Record Indicator 12

CMS Update Task 12

UpdateFlag API 13
Data Flow 14
Error Handling 15

Ongoing Metadata Synchronization 16

Orphan CMS Records System Job 16

Delivery by SRU (Content Aggregator) 17

SRU Response 17
Content Aggregator 18

Chapter 3 Configuring Rosetta 21

SRU Definitions Configuration 21

Configuring the External Resource File 21
Configuring the Explain File 22

Configuring the CMS Update Task 23

Orphan CMS Records Job 27

Chapter 4 Ongoing Synchronization Process 29

Introduction 29

Output File Naming Convention 30

Export File Content 31

Error Handling 32

1

Overview of Rosetta-Voyager Synchronization

Voyager is used in the context of the Rosetta as a Collection Management System. As such, descriptive records in Voyager are used to describe sets of Intellectual Entities (IE) in Rosetta.

In the cases where there are objects that are cataloged (such as described as part of a collection) in the Voyager system, the collection descriptive record should also be preserved, necessitating a full synchronization between the metadata in both systems.

In order to keep the data in the Rosetta updated, any changes made to the Collection Metadata record in Voyager can trigger a synchronization of the updated record in Rosetta.

To support the synchronization, there must be a link between a Collection Descriptive record in Voyager and the set of related objects in Rosetta. This is achieved by storing the CMS record ID (such as the system number) as one of the attributes of the object.

In addition, the linkage between the Collection Descriptive record in Voyager and the objects in Rosetta will allow end users to search the Discovery system of Voyager for relevant collection records and based on the results decide to deliver content using delivery services provided by Rosetta.

2

CMS Integration

This section includes:

- **Introduction** on page **7**
- **Management of the CMS ID** on page **8**
- **Searching and Retrieving Records in CMS Through SRU Protocol** on page **8**
- **CMS Record Indicator** on page **12**
- **CMS Update Task** on page **12**
- **Ongoing Metadata Synchronization** on page **16**
- **Orphan CMS Records System Job** on page **16**
- **Delivery by SRU (Content Aggregator)** on page **17**

Introduction

Rosetta-Voyager synchronization supports the following scenarios:

1. Manual search in CMS by record title (using the SRU protocol) and assignment of a CMS record to an IE in Rosetta.
2. Automatic pre-ingest assignment of a CMS record ID to an IE.
3. Delivery by SRU request (search in Rosetta by CMS ID) and reception of the Delivery URLs of the requested IEs that share the same CMS ID.

All scenarios support the following features:

- The CMS data is replicated and stored in Rosetta as a CMS record.
- The CMS record can be updated based on updates that arrive from CMS. The records from CMS must be in OAI DC format.
- The record in CMS has an indication that it is associated with IEs in Rosetta.

Management of the CMS ID

The integration between Rosetta and Voyager is based on the CMS ID. This identifier is unique to each record in Voyager but can be shared in Rosetta by multiple IEs. (Only IE can be associated with a CMS record, not a representation or a file)

The CMS ID is the record identifier of the Voyager record (system number).

Searching and Retrieving Records in CMS Through SRU Protocol

Rosetta can use the SRU protocol for searching CMS either by the title of the record (when searching manually) or by the ID (when the ID is populated in the METS XML file, pre-ingest). The search by title is done online by users who select only one record from the list of possible matching records.

The search by ID is done by the system and only one record is expected in the SRU response.

Search by Title (Manual Assignment)

The syntax of the SRU request when searching by title is as follows:

```
http://<sru-
server>?version=1.1&operation=searchRetrieve&query=<title>&
maximumRecords=1&recordSchema=dc
```

The following are the parameters of the SRU request when searching by title.

- ***<SRU-server>*** – holds the SRU address of the CMS as it is written in the SRU configuration XML
- **version** – the version of the request and a statement by the client that it wants the response to be less than, or preferably equal to, this version (should be 1.1)
- **operation** – the string `searchRetrieve`.
- **query** – holds the title of the searched CMS record
- **maximumRecords** – the limit on record returned (for example, only 10 records are returned when searching by title)
- **recordSchema** – should always be **dc** for retrieving the record in Dublin Core format

The following is an example of an SRU request:

```
http://z3950.loc.gov:7090/
aleph?version=1.1&operation=searchRetrieve&query=dinosaur&max
imumRecords=1&recordSchema=dc
```

Data Flow

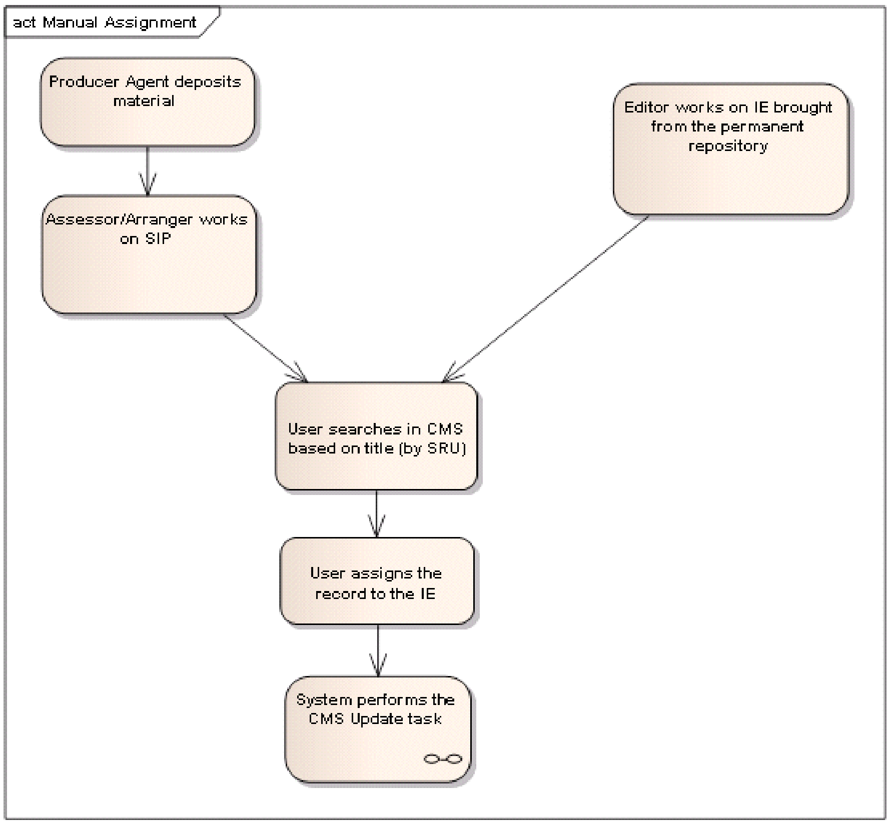

act Manual Assignment
Producer Agent deposits material
Assessor/Arranger works on SIP
Editor works on IE brought from the permanent repository
User searches in CMS based on title (by SRU)
User assigns the record to the IE
System performs the CMS Update task

Figure 1: Manaul Assignment Data Flow

Search by ID

The syntax of the SRU request when searching by ID is as follows:

```
http://<sru-
server>?version=1.2&operation=searchRetrieve&query=<id>&
maximumRecords=1&recordSchema=dc
```

The following are the parameters of the SRU request when searching by ID:

- *<SRU-server>* – holds the SRU address of the CMS as it is written in the SRU configuration XML

- **version** – the version of the request and a statement by the client that it wants the response to be less than, or preferably equal to, this version (should be 1.2)
- **operation** – the string `searchRetrieve`
- **query** – holds the ID of the searched record as it is populated in the METS by the submission application
- **maximumRecords** – should be limited to one record since there should be no more than one BIB record with this ID
- **recordSchema** – should always be **dc** for retrieving the record in Dublin Core format

NOTE:
The ID that Rosetta is searching by is the ID provided in the METS file.

Data Flow

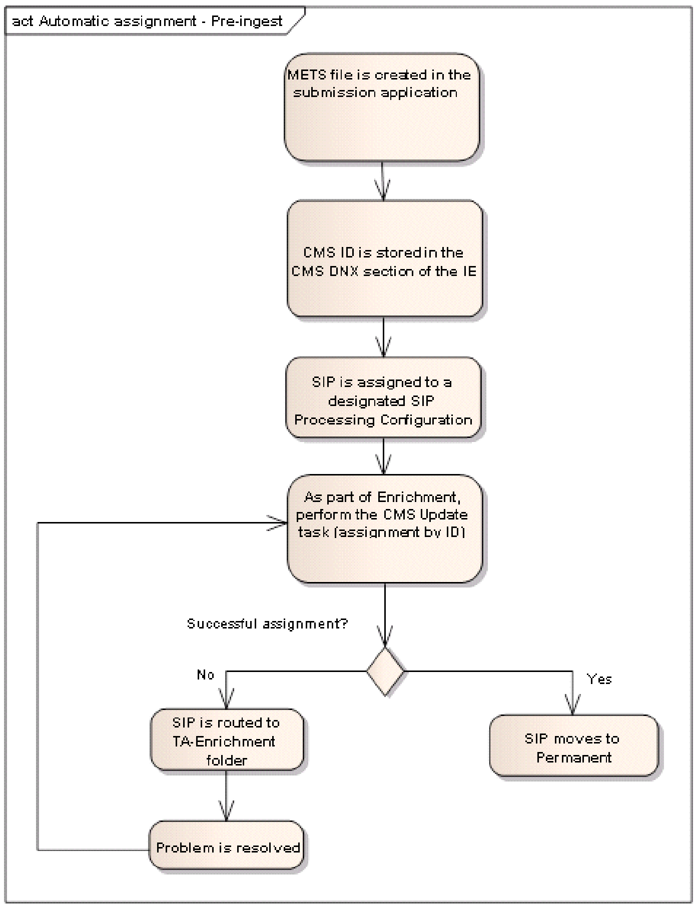

act Automatic assignment - Pre-ingest
METS file is created in the submission application
CMS ID is stored in the CMS DNX section of the IE
SIP is assigned to a designated SIP Processing Configuration
As part of Enrichment, perform the CMS Update task (assignment by ID)
Successful assignment?
No
Yes
SIP is routed to TA-Enrichment folder
SIP moves to Permanent
Problem is resolved

Figure 2: Pre-Ingest - Automatic Assignment Data Flow

SRU Response

The SRU response when searching by title is an XML that contains the DC records of the matching CMS records. Rosetta displays the list of records and allows the user to choose only one record that becomes the CMS record.

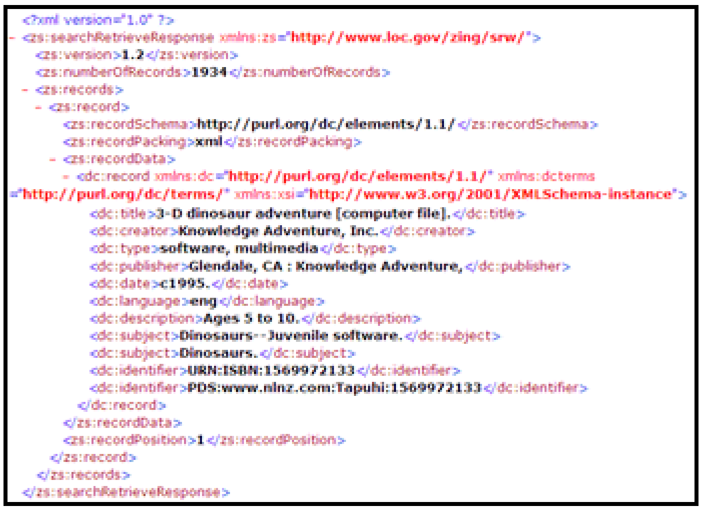
The CMS ID that will be used by Rosetta is held in the **dc:identifier** field:

```
<dc:identifier>DPS:<machine name>:<library code>:<System number></
dc:identifier>
```

For example:

```
<dc:identifier>DPS: 10.1.234.165:voyager:000056929</dc:identifier>
```

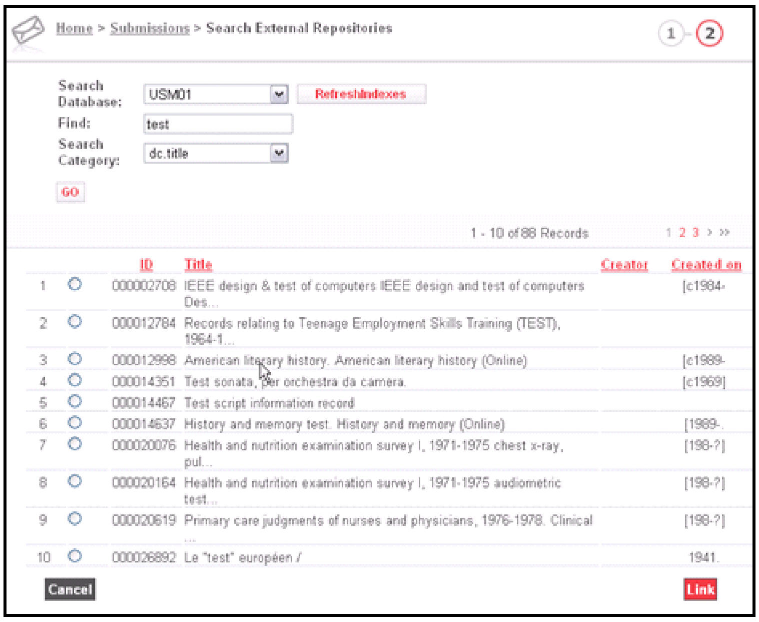
The following is an example of the resulting SRU response in XML format:



```
  <?xml version="1.0" ?>
- <zs:searchRetrieveResponse xmlns:zs="http://www.loc.gov/zing/srw/">
    <zs:version>1.2</zs:version>
    <zs:numberOfRecords>1934</zs:numberOfRecords>
  - <zs:records>
    - <zs:record>
        <zs:recordSchema>http://purl.org/dc/elements/1.1/</zs:recordSchema>
        <zs:recordPacking>xml</zs:recordPacking>
      - <zs:recordData>
        - <dc:record xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcterms
="http://purl.org/dc/terms/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <dc:title>3-D dinosaur adventure [computer file].</dc:title>
            <dc:creator>Knowledge Adventure, Inc.</dc:creator>
            <dc:type>software, multimedia</dc:type>
            <dc:publisher>Glendale, CA : Knowledge Adventure,</dc:publisher>
            <dc:date>c1995.</dc:date>
            <dc:language>eng</dc:language>
            <dc:description>Ages 5 to 10.</dc:description>
            <dc:subject>Dinosaurs--Juvenile software.</dc:subject>
            <dc:subject>Dinosaurs.</dc:subject>
            <dc:identifier>URN:ISBN:1569972133</dc:identifier>
            <dc:identifier>PDS:www.nlnz.com:Tapuhi:1569972133</dc:identifier>
          </dc:record>
        </zs:recordData>
        <zs:recordPosition>1</zs:recordPosition>
      </zs:record>
    </zs:records>
  </zs:searchRetrieveResponse>
```

Figure 3: SRU Response in XML Format

The following example shows an SRU response display in Rosetta (in manual assignment):



Search Database: USM01

Find: test

Search Category: dc.title

1 - 10 of 88 Records 1 2 3 > >>

		ID	Title	Creator	Created on
1	○	000002708	IEEE design & test of computers IEEE design and test of computers Des...		[c1984-
2	○	000012784	Records relating to Teenage Employment Skills Training (TEST), 1964-1...		
3	○	000012998	American literary history. American literary history (Online)		[c1989-
4	○	000014351	Test sonata, per orchestra da camera.		[c1969]
5	○	000014467	Test script information record		
6	○	000014637	History and memory test. History and memory (Online)		[1989-.
7	○	000020076	Health and nutrition examination survey I, 1971-1975 chest x-ray, pul...		[198-?]
8	○	000020164	Health and nutrition examination survey I, 1971-1975 audiometric test...		[198-?]
9	○	000020619	Primary care judgments of nurses and physicians, 1976-1978. Clinical ...		[198-?]
10	○	000026892	Le "test" européen /		1941.

Figure 4: SRU Response Display in Rosetta (Manual Assignment)

NOTE:
The response for the search by ID request can only be one record. Rosetta will use this record as the CMS record.

CMS Record Indicator

The ExistsInPreservationSystem flag in Voyager marks descriptive records in the Voyager system as being associated with objects in Rosetta. If this flag is set to True, the WEB-OPAC system (Timeframes) will give end users the option to retrieve the objects from Rosetta via SRU.

CMS Update Task

The system uses CMS Update task to manage the creation of the CMS records in Rosetta, the generation and assignment of their IDs, and the update of the matching IE in Rosetta and the Voyager record. This task performs the following actions:

1 If the DC XML is available (retrieved by the SRU request in the first scenario – see **Introduction** on page 7), the task takes the value of the

`dc:identifier` record. (For the second scenario, this ID is already stored in the CMS section of the METS XML.)

2. Checks Rosetta (in the HDEMETADATA table) for an existing CMS record that matches the identifier.
   - If there is already a matching CMS record, the CMS ID is stored in the CMS section in the DNX of the processed IE and the task is completed.
   - If there is no such record in Rosetta, an SRU request is initiated for obtaining the record from Voyager by the Rosetta ID.
3. A CMS record is created in Rosetta, and the CMS ID, as it arrived in the SRU response, is the record identifier.

To update the CMS, the UpdateFlag API is called by Rosetta to update records.

UpdateFlag API

The Update Flag API in Voyager uses the following parameters:

- a string that represents the ID of the object that needs to be synchronized
- a boolean flag that determines whether the flag should be on or off (as it is stored in Rosetta)

The API is implemented as an HTTP request from Rosetta, as follows:

```
http://<base-url>?updateFlag=<flag>&cmsId=<recordId>
```

For example:

```
http://z3950.loc.gov:7090/voyager/ctx?updateFlag=true&cmsId=500
```

The HTTP response contains a success flag for a successful update of a CMS record or a general error description, if such occurred.

For general errors, the response XML (where the Mime Type should be set to **application/xml**) should look similar to the following example:

:

```
<update_exist_in_preservation_flag_output>
        <error>failed connecting to DB</error>
</update_exist_in_preservation_flag_output>
```

If no general error occurs, the output includes the status of the update operation – `success` or `error`. The following description is displayed:

```
<update_exist_in_preservation_flag_output>
        <cms id="1234" status="success"/>
</update_exist_in_preservation_flag_output>
```

or

```
<update_exist_in_preservation_flag_output>
        <cms id="1234" status="error">record "1234" does not exist
in Aleph</cms>
</update_exist_in_preservation_flag_output>
```

Data Flow

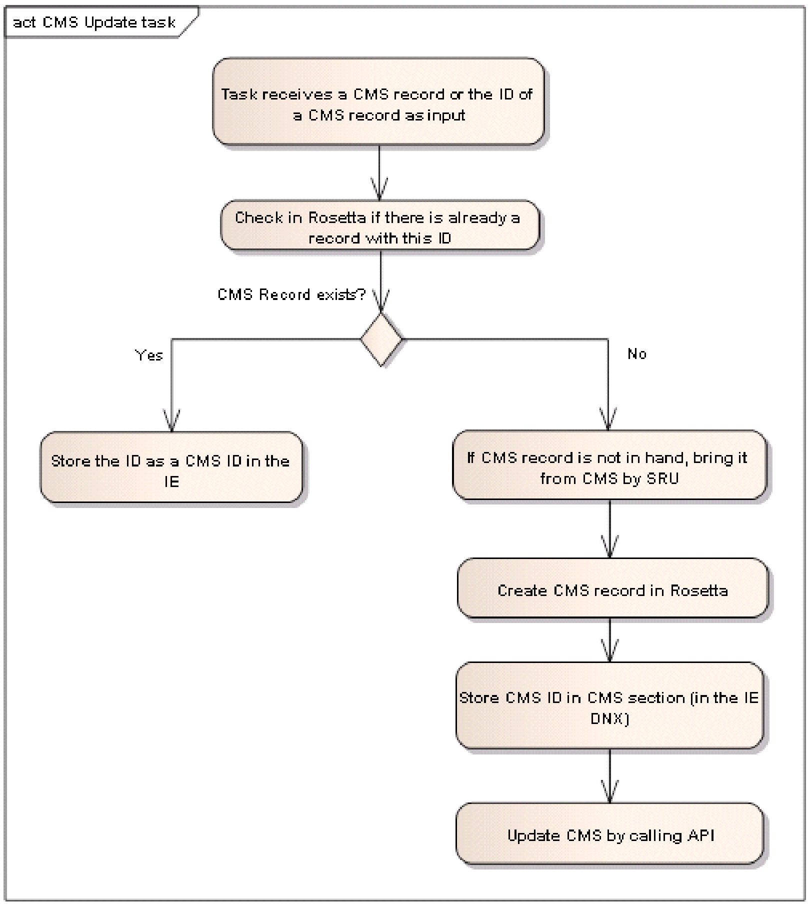

act CMS Update task
Task receives a CMS record or the ID of a CMS record as input
Check in Rosetta if there is already a record with this ID
CMS Record exists?
Yes
No
Store the ID as a CMS ID in the IE
If CMS record is not in hand, bring it from CMS by SRU
Create CMS record in Rosetta
Store CMS ID in CMS section (in the IE DNX)
Update CMS by calling API

Figure 5: CMS Update Task Data Flow

Error Handling

The following errors might occur during synchronization tasks:

Table 1. Error Handling

Error Name	Description	Solution
No connection between CMS and Rosetta.	Could be due to wrong SRU configuration or a network problem.	System Administrator should make sure that: ■ the SRU XML configuration file is correct (see Configuring Rosetta on page 21). ■ the network configuration allows requests in and out of Rosetta (such as firewall definitions).
Record was not found in CMS.	The CMS record was not found when the user searched for it by title.	Make sure the searched word is part of the record's title and retry.
The CMS record in Rosetta was not created.	The CMS record was not created. The task aborts.	Manual assignment – Retry after resolving the issue. Automatic assignment – The SIP is moved to the Technical Analyst - Enrichment folder. The Technical Analyst should rerun the process after resolving the issue.

Error handling is different for each of the following scenarios:

- **Manual assignment** – When the CMS record is assigned manually by an Assessor/Arranger or an Editor, the assignment is done online, so that any issue with connecting to Voyager or creating the CMS record in Rosetta will display in an error message on the user's screen. The user will have to make sure that the problem is resolved before trying again.
- **Automatic assignment (Pre-ingest)** – If the IE already has a populated DNX section with a CMS ID in it, the system will perform the CMS Update task as part of the Enrichment phase. If there are errors (such as a record could not be found in Voyager, a connection to Voyager could not be found, or Rosetta could not create a CMS record), the SIP will be directed to the TA – Enrichment folder where these issues can be investigated. The IE will not continue to the "Move to Permanent" phase before this issue is resolved.

Ongoing Metadata Synchronization

Rosetta's main goal is to physically preserve digital content (such as images, video files, audio files, and so forth).

Preserving the metadata that is managed in Voyager is not one of Rosetta's responsibilities. The reason that CMS records are kept in Rosetta is twofold:

- The CMS metadata is sometimes used as collection level metadata. When the same description is relevant for multiple IEs and for efficient management of sets, IEs can be grouped together as related to the same CMS record. For example, each e-journal article is represented as an IE in Rosetta, and the information about the e-journal itself is stored in one CMS record instead of being copied for all of the IEs.
- For delivery purposes, it is possible to show the IE with the CMS metadata, even though the best practice is to start the delivery from the Voyager side (for example, Timeframes) and use Rosetta's Delivery module for retrieving only the stream files, without the descriptive metadata.

For record deletions in Voyager, the CMS record in Rosetta needs to be indicated as deleted as well. The indication is done by setting the **status** field of the CMS record to **deleted**, without changing the record or updating the IEs that are associated with this CMS record. (Disassociate the IEs manually.)

To support these requirements, Rosetta must allow the CMS metadata to be updated on an ongoing basis to reflect the changes that occur in Voyager.

There are no definite business rules regarding the frequency of such synchronization between the systems, which may not even be run by some Rosetta users. Therefore, the current method is not to use the standard Publishing platform in Rosetta (using OAI-PMH and DB tables), but to use a basic method of loading files that contain the updated records published from Voyager.

For a detailed specification of this process in Rosetta, see **Ongoing Synchronization Process** on page **29**.

Orphan CMS Records System Job

Once a day, Rosetta performs a process that scans all of the CMS records that are no longer associated with any IE in Rosetta.

Rosetta uses the UpdateFlag API for these CMS records. Because these records are no longer associated with any IE, the Yes/No indicator of the Rosetta field is set to No. For more information on this API, see **UpdateFlag API** on page **13**.

Delivery by SRU (Content Aggregator)

Rosetta provides an SRU API that returns a list of IEs that share the same CMS ID, with the CMS ID as the query term.

This API is called from the resource discovery system used by Voyager (such as Timesframe), queries Rosetta for the list of IEs, and displays them to the end user.

The syntax of the SRU request is as follows:

```
http://<hostname>/delivery/sru?version=1.2&operation=
searchRetrieve&query=system=<repository-code>&recordId=<recordId>
```

Where the parameters are the following:

- **system** – the repository code of the CMS that is linked to Rosetta, which is written in the SRU configuration XML
- **recordId** – the CMS ID that connects the IE and the CMS record

For example:

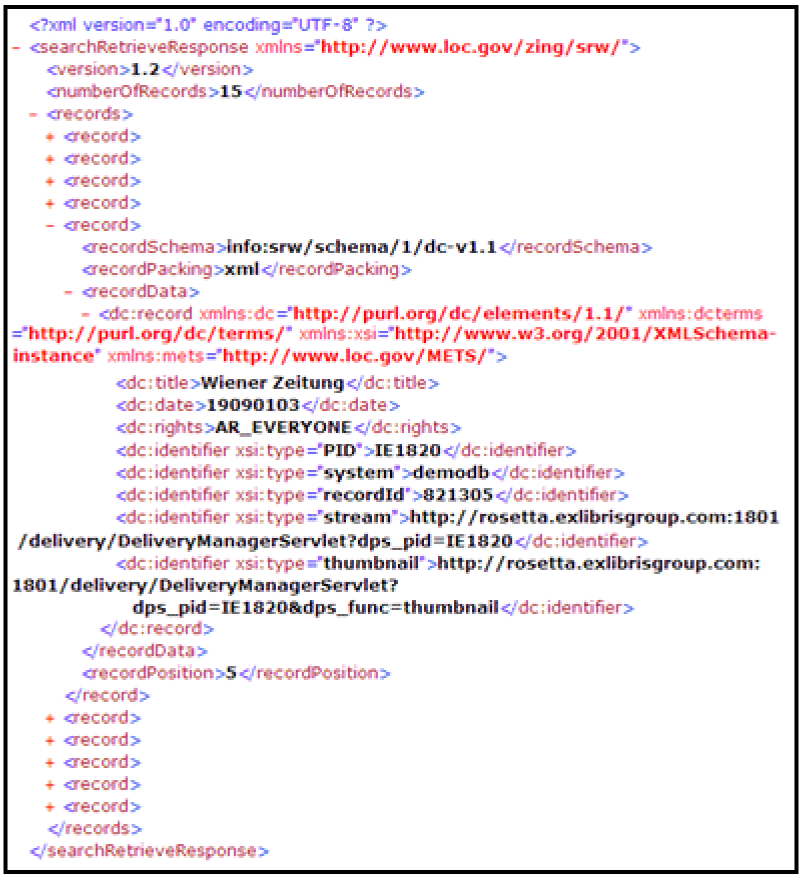
```
http://il-dtldev08c.corp.exlibrisgroup.com:1801/delivery/
sru?version=1.2&operation=searchRetrieve&query=system=demodb&record
Id=13525
```

SRU Response

The SRU response schema includes the descriptive metadata of the IE (in Dublin Core format) and a list of identifiers, differentiated by `xsi:type`. Each `dc:identifier xsi:type` stores one of the following attributes:

- **PID** – the Rosetta ID of the IE.
- **system** – the name of the CMS that is linked to Rosetta (such as CMS).
- **recordId** – the CMS ID that connects the IE and the CMS record.
- **stream** – the deep link to the IE. This link enables the user to view the IE in Rosetta's Delivery module.
- **thumbnail** – the link to the IE's thumbnail. (If an IE has multiple files, this is the thumbnail of the first file.)

Because a CMS ID can be linked to more than one IE and there is no limit to how many IEs can be linked to one CMS ID, Rosetta returns, in response to the SRU request, an XML file that contains the requested records along with the total number of records. This XML can be accessed by the calling system to perform activities such as sorting and searching for a specific IE.



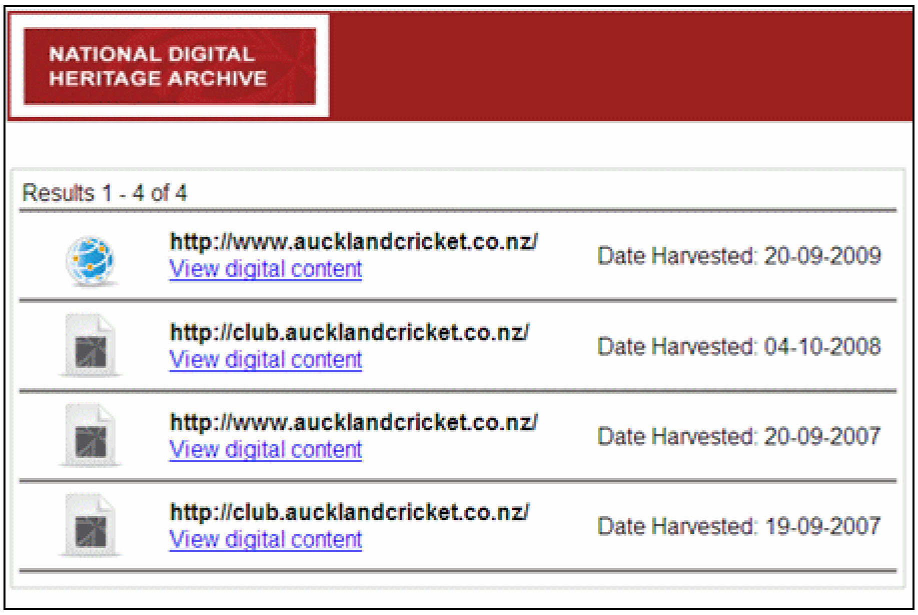
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <searchRetrieveResponse xmlns="http://www.loc.gov/zing/srw/">
    <version>1.2</version>
    <numberOfRecords>15</numberOfRecords>
  - <records>
    + <record>
    + <record>
    + <record>
    + <record>
    - <record>
        <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
        <recordPacking>xml</recordPacking>
      - <recordData>
        - <dc:record xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcterms
="http://purl.org/dc/terms/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:mets="http://www.loc.gov/METS/">
            <dc:title>Wiener Zeitung</dc:title>
            <dc:date>19090103</dc:date>
            <dc:rights>AR_EVERYONE</dc:rights>
            <dc:identifier xsi:type="PID">IE1820</dc:identifier>
            <dc:identifier xsi:type="system">demodb</dc:identifier>
            <dc:identifier xsi:type="recordId">821305</dc:identifier>
            <dc:identifier xsi:type="stream">http://rosetta.exlibrisgroup.com:1801
/delivery/DeliveryManagerServlet?dps_pid=IE1820</dc:identifier>
            <dc:identifier xsi:type="thumbnail">http://rosetta.exlibrisgroup.com:
1801/delivery/DeliveryManagerServlet?
              dps_pid=IE1820&dps_func=thumbnail</dc:identifier>
          </dc:record>
        </recordData>
        <recordPosition>5</recordPosition>
      </record>
    + <record>
    + <record>
    + <record>
    + <record>
    + <record>
    </records>
  </searchRetrieveResponse>
```

Figure 6: SRU Response

Content Aggregator

Because a CMS ID can be linked to more than one IE and there is no limit to how many IEs can be linked to one CMS ID, Rosetta returns, in response to the SRU request, an XML file that contains the requested records along with the total number of records. This XML can be accessed by the calling system to perform activities such as sorting and searching for a specific IE.

If the CMS stores the description of a Web site and the XML returns the list of various harvests of this Web site, the calling application can sort the harvests based on dates so that the user can select the exact IE (harvest) to view. For example:


NATIONAL DIGITAL HERITAGE ARCHIVE

Results 1 - 4 of 4

http://www.aucklandcricket.co.nz/ View digital content	Date Harvested: 20-09-2009
http://club.aucklandcricket.co.nz/ View digital content	Date Harvested: 04-10-2008
http://www.aucklandcricket.co.nz/ View digital content	Date Harvested: 20-09-2007
http://club.aucklandcricket.co.nz/ View digital content	Date Harvested: 19-09-2007

Figure 7: Harvest Sorted by Date

NOTE:
The Content Aggregator was developed by the National Library of New Zealand and is available to all Rosetta users.

3

Configuring Rosetta

This section describes the items that need to be configured in Rosetta in order to set up a connection between Rosetta and Voyager.

This section includes:

- SRU Definitions Configuration on page 21
- Configuring the CMS Update Task on page 23
- SRU Definitions Configuration on page 21

SRU Definitions Configuration

The following tasks must be performed in the configuration of SRU:

- Configure the external resource file
- Configure the explain file

Configuring the External Resource File

To configure the external resource file:

1. From the Administration home page, go to the **Advanced Configuration > General > Configuration Files** page.
2. In the **File Group** field, select **External Interface**.
3. In the **Sub-Group** field, select **SRU/SRW**. The page reloads with entries that reflect your selections.
4. Click **Edit** for the file named `external_resource_explorer_configuration.xml`. The file opens. It contains all SRU configurations that are delivered with Rosetta.
5. Add your configuration to the file. Modify the following parameters:
   - **baseUrl** – The URL of the system that acts as an SRU server (CMS)

- **version** – SRU version (currently 1.2 is supported by Rosetta and Voyager)
- **operation** – Since Rosetta is the client, **searchRetrieve** is the only option.
- **recordSchema** – Set to **dps**
- **indexName** – Set to **rec.id** (system number index in Voyager)
- **recordPacking** – Set to **xml**
- **updateUrl** – If the server can get a response for updating the **Exist in Rosetta** flag, add the URL here. This is currently available only for Voyager and Aleph. For other systems, leave this field empty.

The following example shows an SRU configuration for Voyager:

```
<?xml version="1.0" encoding="UTF-8"?>
<ExternalResourceExplorer
xmlns="http://www.loc.gov/zing/srw/configuration/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xml="http://www.w3.org/XML/1998/namespace"
>
<RepositoryName name="qa722mkdb" >
<protocol type="SRW">
           <parm name="baseUrl">http://10.1.234.165:54091/voyager</
parm>
            <parm name="version">1.1</parm>
            <parm name="operation">searchRetrieve</parm>
            <parm name="recordSchema">dps</parm>
            <parm name="indexName">rec.id=</parm>
            <parm name="recordPacking">xml</parm>
            <parm name="updateUrl">http://10.1.234.165:54091/vxws/
updateDPSFlag</parm>
 </protocol>
 </RepositoryName>
```

Configuring the Explain File

To configure the explain file:

1. From the Administration home page, go to **Advanced Configuration > General > Configuration Files**.
2. For the **File Group** field, select **External Interface**.
3. For the **Sub-Group**, select **SRU/SRW**. The page reloads with entries that reflect your selections.
4. Click **Edit** for the file named `explain.properties` and enter your SRU (Rosetta) details:
   - host={host}
   - port={port}

- database={repository schema}
- dbInfo={DPS SRU Database}
- numberOfRecords={x}

For example:

```
host=dps1.corp.exlibrisgroup.com
port=1801
database=V212_rep00
dbInfo=DPS SRU Database
numberOfRecords=0
```

Explain Response XML

The URL for reviewing the explain response XML is `http://<host-name>:1801/core-delivery/sru?operation=explain`.

For example:

```
http://il-dps02.corp.exlibrisgroup.com:1801/core-delivery/
sru?operation=explain
```

NOTE:
The content of this XML is not configurable. It includes the information regarding the server, the database, and the indexed DC fields.

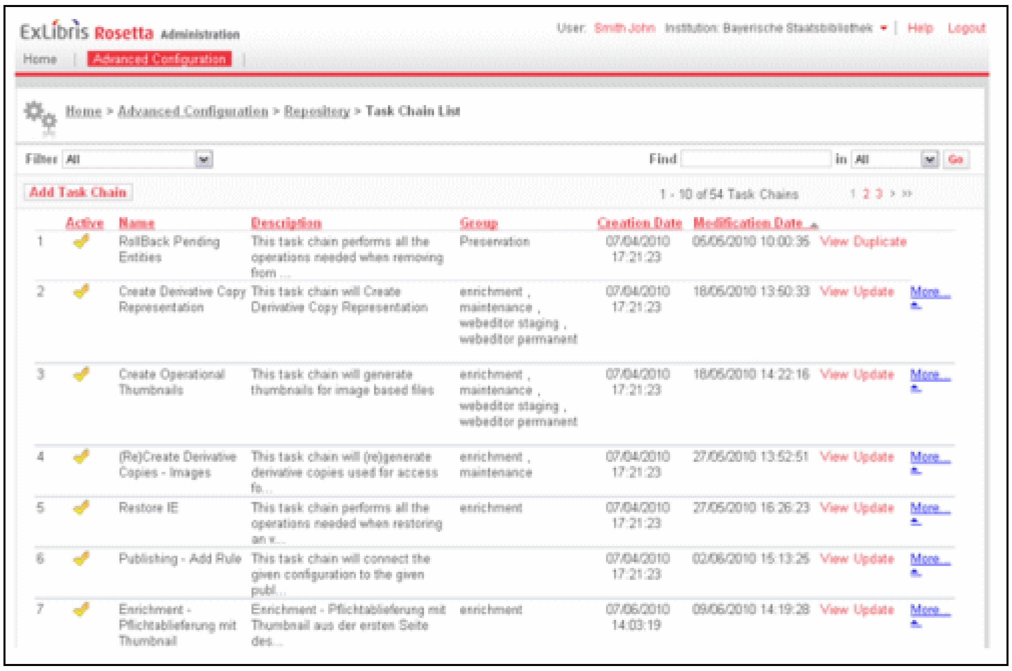
Configuring the CMS Update Task

When a CMS ID is assigned to an IE, Rosetta verifies that the copy of the CMS record is held locally, as part of the shared metadata. This synchronization task is processed as part of the Enrichment task chain in Rosetta.

The same task is also used when the user performs the **Assign CMS** action in the Web editor to an IE that is already in permanent.

To ensure that the CMS Update task is part of the enrichment task chain:

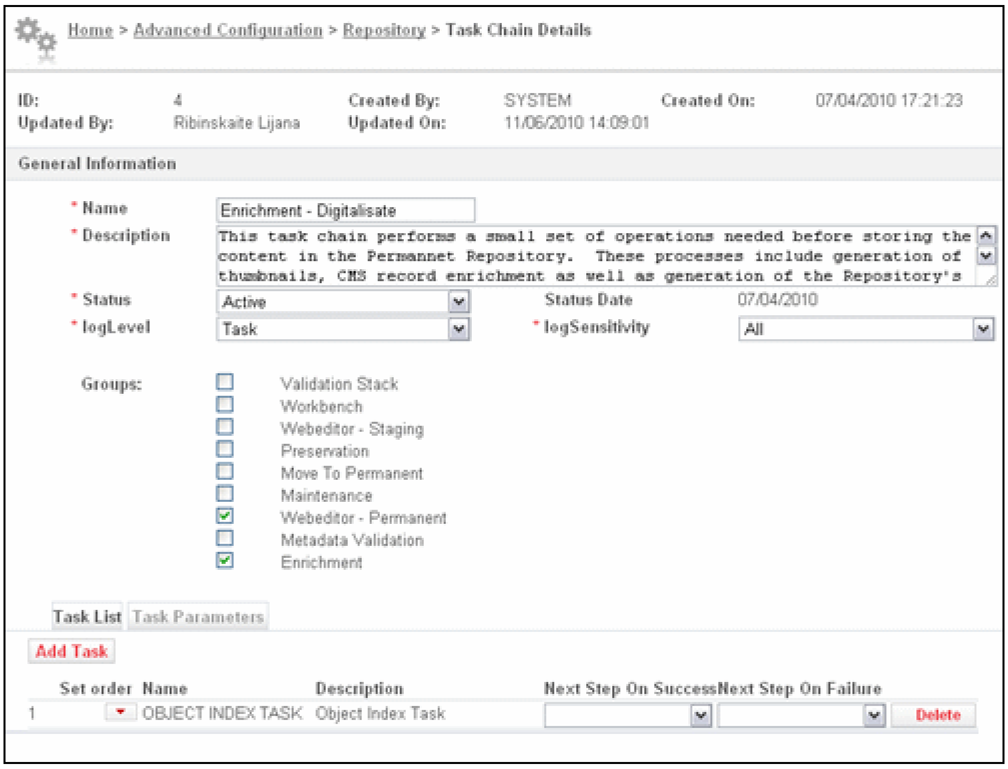
1 From the Administration home page, go to **Advanced Configuration > Repository > List of Task Chains**. A list of task chains appears.


ExLibris Rosetta Administration
User: Smith John
Institution: Bayerische Staatsbibliothek
Help
Logout
Home
Advanced Configuration
Home > Advanced Configuration > Repository > Task Chain List
Filter: All
Find
in All
Go
Add Task Chain
1 - 10 of 54 Task Chains
1 2 3 > >>

	Active	Name	Description	Group	Creation Date	Modification Date		
1	✓	RollBack Pending Entities	This task chain performs all the operations needed when removing from ...	Preservation	07/04/2010 17:21:23	05/05/2010 10:00:35	View Duplicate	
2	✓	Create Derivative Copy Representation	This task chain will Create Derivative Copy Representation	enrichment , maintenance , webeditor staging , webeditor permanent	07/04/2010 17:21:23	18/05/2010 13:50:33	View Update	More...
3	✓	Create Operational Thumbnails	This task chain will generate thumbnails for image based files	enrichment , maintenance , webeditor staging , webeditor permanent	07/04/2010 17:21:23	18/05/2010 14:22:16	View Update	More...
4	✓	(Re)Create Derivative Copies - Images	This task chain will (re)generate derivative copies used for access fo...	enrichment , maintenance	07/04/2010 17:21:23	27/05/2010 13:52:51	View Update	More...
5	✓	Restore IE	This task chain performs all the operations needed when restoring an v...	enrichment	07/04/2010 17:21:23	27/05/2010 16:26:23	View Update	More...
6	✓	Publishing - Add Rule	This task chain will connect the given configuration to the given publ...		07/04/2010 17:21:23	02/06/2010 15:13:25	View Update	More...
7	✓	Enrichment - Pflichtablieferung mit Thumbnail	Enrichment - Pflichtablieferung mit Thumbnail aus der ersten Seite des...	enrichment	07/06/2010 14:03:19	09/06/2010 14:19:28	View Update	More...

Figure 8: List of Task Chains

2 Select the Enrichment task chain for which you want to configure the CMS Update task and click **Update**. The Task Chain Details page opens.



Home > Advanced Configuration > Repository > Task Chain Details

ID:	4	Created By:	SYSTEM	Created On:	07/04/2010 17:21:23
Updated By:	Ribinskaite Lijana	Updated On:	11/06/2010 14:09:01		

General Information

* Name	Enrichment - Digitalisate		
* Description	This task chain performs a small set of operations needed before storing the content in the Permannet Repository. These processes include generation of thumbnails, CMS record enrichment as well as generation of the Repository's		
* Status	Active	Status Date	07/04/2010
* logLevel	Task	* logSensitivity	All

Groups:
- [ ] Validation Stack
- [ ] Workbench
- [ ] Webeditor - Staging
- [ ] Preservation
- [ ] Move To Permanent
- [ ] Maintenance
- [x] Webeditor - Permanent
- [ ] Metadata Validation
- [x] Enrichment

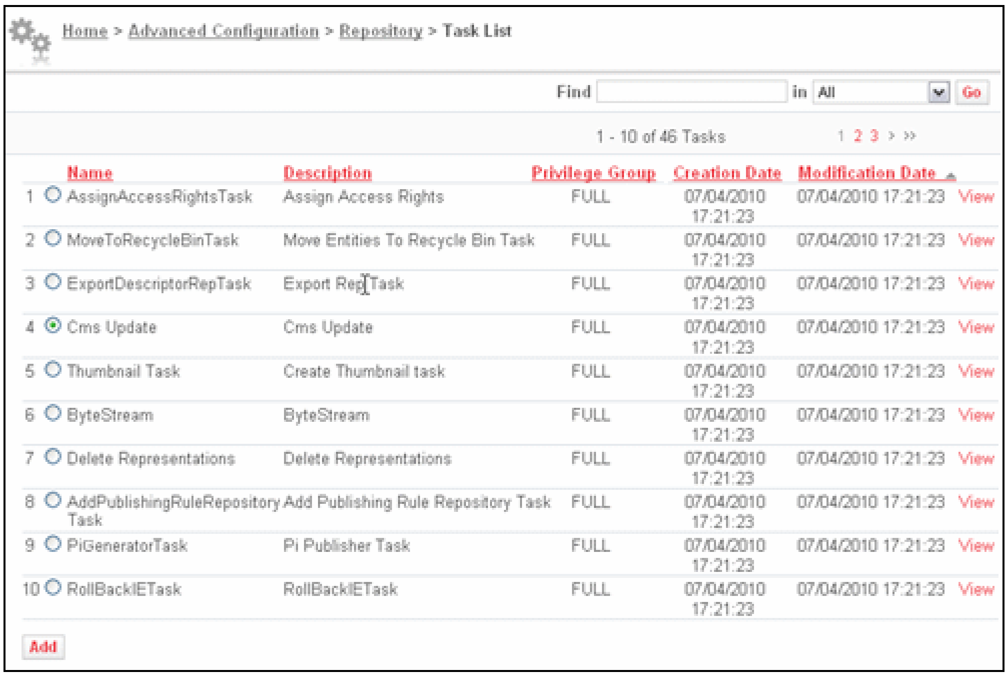
Task List | Task Parameters

Add Task

Set order	Name	Description	Next Step On Success	Next Step On Failure	
1	OBJECT INDEX TASK	Object Index Task			Delete

Figure 9: Task Chain Details

3 Click the **Add Task** button and select **CMS Update** from the list of tasks.



Home > Advanced Configuration > Repository > Task List

Find [ ] in All Go

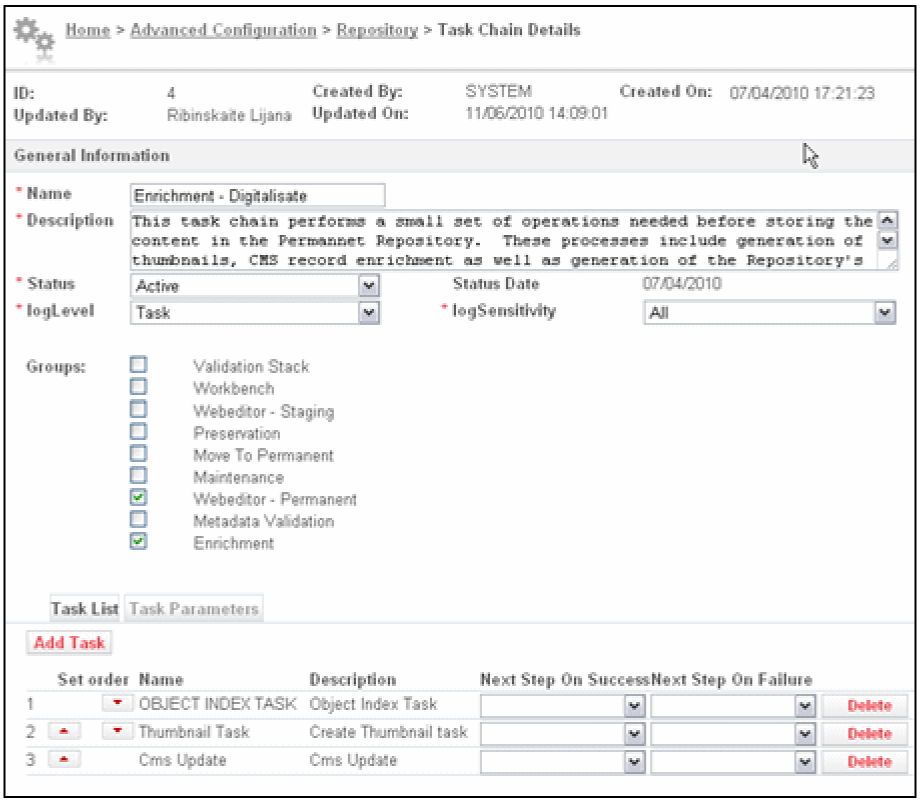
1 - 10 of 46 Tasks   1 2 3 > >>

		Name	Description	Privilege Group	Creation Date	Modification Date	
1	○	AssignAccessRightsTask	Assign Access Rights	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View
2	○	MoveToRecycleBinTask	Move Entities To Recycle Bin Task	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View
3	○	ExportDescriptorRepTask	Export Rep Task	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View
4	◉	Cms Update	Cms Update	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View
5	○	Thumbnail Task	Create Thumbnail task	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View
6	○	ByteStream	ByteStream	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View
7	○	Delete Representations	Delete Representations	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View
8	○	AddPublishingRuleRepository Task	Add Publishing Rule Repository Task	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View
9	○	PiGeneratorTask	Pi Publisher Task	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View
10	○	RollBackIETask	RollBackIETask	FULL	07/04/2010 17:21:23	07/04/2010 17:21:23	View

Add

Figure 10: Adding the CMS Update Task

The Enrichment task chain now includes the CMS Update task.



Home > Advanced Configuration > Repository > Task Chain Details

ID:	4	Created By:	SYSTEM	Created On:	07/04/2010 17:21:23
Updated By:	Ribinskaite Lijana	Updated On:	11/06/2010 14:09:01		

General Information

* Name	Enrichment - Digitalisate		
* Description	This task chain performs a small set of operations needed before storing the content in the Permannet Repository. These processes include generation of thumbnails, CMS record enrichment as well as generation of the Repository's		
* Status	Active	Status Date	07/04/2010
* logLevel	Task	* logSensitivity	All

Groups:
- ☐ Validation Stack
- ☐ Workbench
- ☐ Webeditor - Staging
- ☐ Preservation
- ☐ Move To Permanent
- ☐ Maintenance
- ☑ Webeditor - Permanent
- ☐ Metadata Validation
- ☑ Enrichment

Task List | Task Parameters

Add Task

Set order	Name	Description	Next Step On Success	Next Step On Failure	
1	OBJECT INDEX TASK	Object Index Task			Delete
2	Thumbnail Task	Create Thumbnail task			Delete
3	Cms Update	Cms Update			Delete

Figure 11: Enrichment Task Chain with CMS Update

Orphan CMS Records Job

The Orphan CMS Records job is automatically configured to run every 24 hours.

NOTE:
There is no UI available in Rosetta to reconfigure this job.

4

Ongoing Synchronization Process

This section includes:

- **Introduction** on page **29**
- **Output File Naming Convention** on page **30**
- **Export File Content** on page **31**
- **Error Handling** on page **32**

Introduction

This section describes the process used for supporting ongoing synchronization.

On the CMS side, the Publishing mechanism is used for exporting records to Rosetta, in the Rosetta DC format, using an `*.xml` file. To publish records that are linked to Rosetta, you must define a base set, including records that are linked to Rosetta objects (base for all records that indicate **Yes** in the Rosetta field).

The records are published in OAI-DC format. Each exported file can contain no more than 1000 records. Each file should be in the OAI-PMH response format, with a record header and record data in **oai_dc** format.

Rosetta retrieves the updated records in the export files and updates the matching CMS records in the HDEMETADATA table. The export files are expected to be in the NFS, in an area that both CMS and Rosetta can access.

The location of this directory is stored in the `metadata_job_config.xml` configuration file, which is accessed from the **Home > Advanced Configuration > General > Configuration Files** page by selecting **External Interfaces** from the **File Group** filter and **External CMS** from the **Sub-Group** filter.

The following is an example of this file:

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata_job_conf
  xmlns="http://com/exlibris/digitool/repository/jobs/xmlbeans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xml="http://www.w3.org/XML/1998/namespace">
<!--
  <synch_job>

<adapter>com.exlibris.digitool.repository.md.FileAdapterForOAI</
adapter >
    <adapter_param key="mddir" value="/exlibris/dtl/d4_2/tmp"></
adapter_param>
    <adapter_param key="processeddir" value="/exlibris/dtl/d4_2/
tmp/processed"></adapter_param>
    <adapter_param key="filenameprefix" value="bvb01"></
adapter_param>
    <adapter_param key="vebose" value="false"></adapter_param>
  </synch_job>
  -->
</metadata_job_conf>
```

Output File Naming Convention

The list of all published CMS records are exported to an `*.xml` file, available for Rosetta under a directory shared by both systems. The file must be broken into smaller and more manageable pieces so that each file does not contain more than 1000 records and each file is a valid OAI-PMH response.

The following is the naming convention for a set of files representing a single extraction:

```
<repositoryCode>.<recordFormat>.<yyMMddhhmmss>.<sequence>.xml
```

Where the repositoryCode is the name of the CMS (for example, Voyager) as it is defined in the SRU configuration XML, the set name is the name of the publishing set, and the date indicates the date of the export.

For example, an export smaller than 1000 records might have a name that is similar to the following name:

```
voyager.oai_dc.080219220123.01.xml
```

For example, a larger export may be broken into multiple files with names similar to the following names:

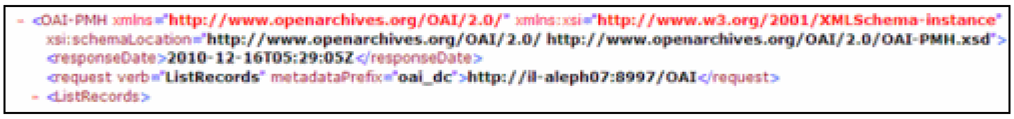
```
voyager.oai_dc.080219220123.01.xml
voyager.oai_dc.080219220123.02.xml
voyager.oai_dc.080219220123.03.xml
```

Files that are not compliant with the above convention are not loaded and ignored.

Export File Content

The structure of the export file should be the structure of an OAI-PMH response. Further instructions on constructing an OAI-PMH response can be found at: http://www.openarchives.org/pmh/

The following is an example of the OAI header:



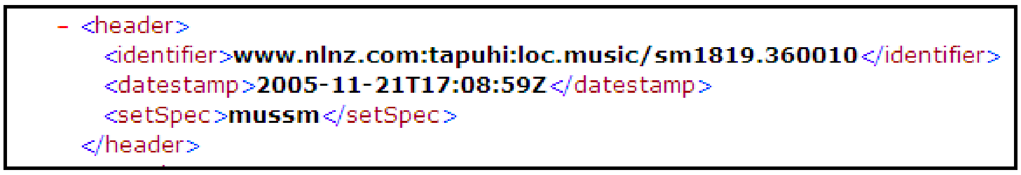
```
- <OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
    <responseDate>2010-12-16T05:29:05Z</responseDate>
    <request verb="ListRecords" metadataPrefix="oai_dc">http://il-aleph07:8997/OAI</request>
  - <ListRecords>
```

Figure 12: OAI Header of the Export File

As with SRU, each record must contain the unique identifier in addition to the descriptive metadata. As mentioned above, the following is the format of the identifier:

```
<identifier>DPS:<machine name>:<library code>:<CMS ID>:
</identifier>
```

For example:



```
  - <header>
      <identifier>www.nlnz.com:tapuhi:loc.music/sm1819.360010</identifier>
      <datestamp>2005-11-21T17:08:59Z</datestamp>
      <setSpec>mussm</setSpec>
    </header>
```

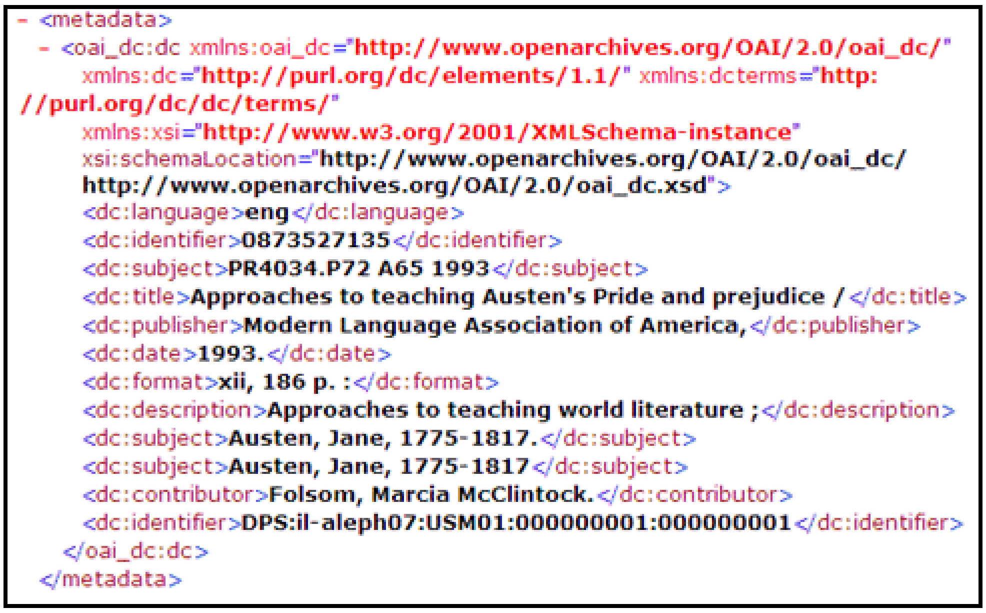
Figure 13: Format of the Identifier

The repository code and the host name must be identical to their definitions in the SRU configuration XML.

If the record is deleted, the status attribute is as follows:

```
<status>"deleted"</status>
```

For updated documents, the DC part of the CMS record is overwritten by the DC section as follows:



```
- <metadata>
  - <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
      xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcterms="http:
//purl.org/dc/dc/terms/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
      http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
      <dc:language>eng</dc:language>
      <dc:identifier>0873527135</dc:identifier>
      <dc:subject>PR4034.P72 A65 1993</dc:subject>
      <dc:title>Approaches to teaching Austen's Pride and prejudice /</dc:title>
      <dc:publisher>Modern Language Association of America,</dc:publisher>
      <dc:date>1993.</dc:date>
      <dc:format>xii, 186 p. :</dc:format>
      <dc:description>Approaches to teaching world literature ;</dc:description>
      <dc:subject>Austen, Jane, 1775-1817.</dc:subject>
      <dc:subject>Austen, Jane, 1775-1817</dc:subject>
      <dc:contributor>Folsom, Marcia McClintock.</dc:contributor>
      <dc:identifier>DPS:il-aleph07:USM01:000000001:000000001</dc:identifier>
    </oai_dc:dc>
  </metadata>
```

Figure 14: DC Section

Error Handling

The process runs automatically on Rosetta's server without any manual launching or monitoring. Therefore, in order to track errors, the System Administrator must find and search the log for any problems that have occurred during the process.