



Rosetta-Aleph Synchronization Guide

Version 6.0.1

CONFIDENTIAL INFORMATION

The information herein is the property of Ex Libris Ltd. or its affiliates and any misuse or abuse will result in economic loss. DO NOT COPY UNLESS YOU HAVE BEEN GIVEN SPECIFIC WRITTEN AUTHORIZATION FROM EX LIBRIS LTD.

This document is provided for limited and restricted purposes in accordance with a binding contract with Ex Libris Ltd. or an affiliate. The information herein includes trade secrets and is confidential.

DISCLAIMER

The information in this document will be subject to periodic change and updating. Please confirm that you have the most current documentation. There are no warranties of any kind, express or implied, provided in this documentation, other than those expressly agreed upon in the applicable Ex Libris contract. This information is provided AS IS. Unless otherwise agreed, Ex Libris shall not be liable for any damages for use of this document, including, without limitation, consequential, punitive, indirect or direct damages.

Any references in this document to third-party material (including third-party Web sites) are provided for convenience only and do not in any manner serve as an endorsement of that third-party material or those Web sites. The third-party materials are not part of the materials for this Ex Libris product and Ex Libris has no liability for such materials.

TRADEMARKS

"Ex Libris," the Ex Libris bridge, Primo, Aleph, Alephino, Voyager, SFX, MetaLib, Verde, DigiTool, Preservation, Rosetta, URM, ENCompass, Endeavor eZConnect, WebVoyage, Citation Server, LinkFinder and LinkFinder Plus, and other marks are trademarks or registered trademarks of Ex Libris Ltd. or its affiliates.

The absence of a name or logo in this list does not constitute a waiver of any and all intellectual property rights that Ex Libris Ltd. or its affiliates have established in any of its products, features, or service names or logos.

Trademarks of various third-party products, which may include the following, are referenced in this documentation. Ex Libris does not claim any rights in these trademarks. Use of these marks does not imply endorsement by Ex Libris of these third-party products, or endorsement by these third parties of Ex Libris products.

Oracle is a registered trademark of Oracle Corporation.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

Microsoft, the Microsoft logo, MS, MS-DOS, Microsoft PowerPoint, Visual Basic, Visual C++, Win32,

Microsoft Windows, the Windows logo, Microsoft Notepad, Microsoft Windows Explorer, Microsoft Internet Explorer, and Windows NT are registered trademarks and ActiveX is a trademark of the Microsoft Corporation in the United States and/or other countries.

Unicode and the Unicode logo are registered trademarks of Unicode, Inc.

Google is a registered trademark of Google Inc.

iPhone is a registered trademark of Apple Inc.

Table of Contents

Chapter 1	Overview of Rosetta-Aleph Synchronization	5
Chapter 2	Rosetta-Aleph Synchronization	7
	Introduction.....	7
	Management of the CMS ID.....	8
	Searching and Retrieving Aleph Records Through SRU Protocol	8
	<i>Search by Title</i>	9
	<i>SRU Response for Searching by Title</i>	9
	<i>Search by ID</i>	11
	Rosetta Field in Aleph Records	12
	CMS Management Tasks	13
	<i>CMS Update Task</i>	13
	<i>Attach API</i>	13
	<i>Copy IE to CMS Task</i>	14
	Ongoing Metadata Synchronization.....	15
	Orphan CMS Records System Job	16
	<i>Detach API</i>	16
	Delivery of Multiple IEs (CMS Resolver).....	16
	Publishing from Rosetta to Aleph.....	17
	<i>Publishing in Rosetta – Overview</i>	18
	<i>Publishing Processes</i>	18
	<i>Manual Publishing Activities</i>	19
	Publishing the URN from Rosetta to Aleph	19
	<i>Scenario 1: Updating the URN in Existing Aleph Records</i>	20
	<i>Scenario 2: URN in Loaded Aleph Records</i>	21
Chapter 3	Error Handling.....	23
Chapter 4	Flow Diagrams.....	25
	CMS Update Task.....	26
	Manual Assignment Scenario	27
	Automatic Assignment – Pre-Ingest	28
	Publishing from Rosetta to Aleph.....	29
	<i>Ingest of a New IE</i>	29

	<i>Publishing Process</i>	30
Chapter 5	Configuring Rosetta	31
	SRU Definitions Configuration	31
	<i>Configuring the External Resource File</i>	31
	<i>Configuring the Explain File</i>	35
	Configuring the CMS Update Task to Be Part of Enrichment	36
	Copying an IE to the CMS.....	39
	<i>CMS ID Generator Plug-in</i>	41
	Orphan CMS Records Job.....	43
	Configuration of the Update URN Task	43
	Configuration of the X-Service URL	44
Chapter 6	Ongoing Synchronization Process	45
	Introduction.....	45
	Output File Naming Convention	46
	Export File Content	47
	Error Handling.....	48
Chapter 7	Configuring Aleph	49
	Overview of Integration Workflow	49
	Publishing Aleph BIB Records to Rosetta	50
	Aleph SRU Server Setup.....	52
	Rosetta CMS Resolver URL.....	52
	Aleph SRU Server 001 Search Setup	53
	Aleph Z39.5 Server Setup	54
	Expand Routines.....	54
	Defining Translation in Aleph of DC XML.....	54
	Defining a Rosetta Link Display in Aleph Web OPAC	55
	Detaching the API Index	55
	Harvesting Records from Rosetta	56
	<i>Aleph OAI Harvester</i>	57
	<i>Aleph rosetta-01 Loader</i>	60
	<i>Format Conversion</i>	61

Overview of Rosetta-Aleph Synchronization

Aleph is used in the context of Rosetta as a Collection Management System (CMS). As such, descriptive records in Aleph are replicated in Rosetta as CMS records, and CMS records are linked to intellectual entities (IEs) by the CMS ID. (The same CMS ID can be linked to one or more IEs. However, each IE can be linked to only one CMS record).

To keep the data in Rosetta updated, changes that are made to the BIB record in Aleph should be replicated in the Rosetta CMS record. Since Rosetta is a preservation system, the frequency of the metadata replication is configurable, subject to an institution's policy.

The connection between the descriptive record in Aleph and the objects in Rosetta allows end users to search Aleph's discovery system for relevant descriptive records and then view the archived content using delivery services provided by Rosetta.

The link between the two systems provides the following functionality:

- The creation of BIB records in Aleph based on harvested metadata from Rosetta
- Ongoing updates and storage of the descriptive record in Rosetta's permanent repository
- Delivery of IEs based on their descriptive record search results in the Aleph discovery interface

2

Rosetta-Aleph Synchronization

This section includes:

- **Introduction** on page 7
- **Management of the CMS ID** on page 8
- **Searching and Retrieving Aleph Records Through SRU Protocol** on page 8
- **Rosetta Field in Aleph Records** on page 12
- **CMS Management Tasks** on page 13
- **Ongoing Metadata Synchronization** on page 15
- **Orphan CMS Records System Job** on page 16
- **Delivery of Multiple IEs (CMS Resolver)** on page 16
- **Publishing from Rosetta to Aleph** on page 17
- **Publishing the URN from Rosetta to Aleph** on page 19

Introduction

Rosetta-Aleph synchronization supports the following scenarios:

- 1 Manual search in Aleph by BIB record title (using the SRU protocol) and assignment of a BIB record to an IE in Rosetta.
- 2 Automatic pre-ingest assignment of a BIB record ID to an IE.
- 3 Publishing of the descriptive metadata of IEs in Rosetta so that Aleph can harvest the metadata and create BIB records for further cataloging.

All scenarios support the following features:

- The BIB record in Aleph is replicated and stored in Rosetta as a CMS record.
- The CMS record can be updated based on updates that arrive from Aleph. The records from Aleph must be in OAI DC format.

- The record in Aleph has an indication that it is associated with a CMS record in Rosetta.
- Rosetta allows calls from resource discovery systems (WEB-OPACs) that are associated with Aleph to search objects in Rosetta (IEs that are associated with the BIB record) according to the SRU protocol.

Management of the CMS ID

For the first two scenarios above, the CMS record is created based on the Aleph BIB record. Therefore, the CMS ID that is stored in the IE is the BIB record ID, generated by Aleph. (The ID can be either the Aleph System Number or the 001 MARC field, depending on the specific implementation).

In the third scenario, in order to support the creation of records in Aleph based on the published metadata of Rosetta, the CMS ID is generated in Rosetta and stored in Aleph, in addition to the existing identifiers of the BIB record.

In all of the scenarios, the CMS ID is sent to Aleph to be stored in the Rosetta field called `ROS`. Based on the ID in this field, resource discovery systems can call Rosetta and retrieve the direct link to the IE that is associated with the CMS ID.

Searching and Retrieving Aleph Records Through SRU Protocol

As described previously, the link between a CMS record and an IE is established by assigning the CMS ID as an attribute of the IE. For further information on the staff-related workflows for this functionality, see the *Rosetta Staff User's Guide*.

SRU version 1.1 is used as the protocol to search and retrieve Aleph records. This protocol uses the `searchRetrieve` operation.

The result of the query is a Dublin Core record. This allows Rosetta to display the metadata record during the search and assignment process.

Rosetta can use the SRU protocol for searching Aleph 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 a user who selects 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

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 Aleph system 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 BIB record
- `maximumRecords` – can be limited (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&maximumRecords=1&recordSchema=dc
```

SRU Response for Searching by Title

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

Rosetta expects the following structure of the `dc:identifier` field:

```
<dc:identifier>DPS:<machine name>:<library code>:<Aleph System number>:<001 MARC field>:<CMS ID>:</dc:identifier>
```

The parameters are sent according to the following logic:

- If the `ROS` field is populated, Aleph uses it and sends it in addition to the Aleph System Number and MARC 001 field.
- If the `ROS` field is empty, Aleph sends the Aleph System Number and 001 field.
- If there is no 001 field, only the Aleph System Number is sent.

- Rosetta uses the last parameter for searching if the CMS record already exists.
- The Aleph System Number is used as a parameter in the Attach API (see [Attach API](#) on page 13).

For example:

```
<dc:identifier>DPS:il-aleph07:USM01:000056929:123456789:</
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 1: SRU Response for Search By Title

The following is an example of the SRU response display in Rosetta:

Home > Submissions > Search External Repositories

Search Database: USM01 Refresh Indexes

Find: test

Search Category: dc.title

GO

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.

Cancel Link

Figure 2: SRU Response in Rosetta

Search by ID

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

```
http://<srp-
server>?version=1.1&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 Aleph system 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 ID of the searched BIB 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. Rosetta assumes that Aleph maintains indexes both for the MARC 001 field and the System Number.

SRU Response for Searching by ID

The response for the search by ID request can be only one BIB record. Rosetta uses this record as the CMS record.

Rosetta Field in Aleph Records

The `ROS` field, which is referred to as the Rosetta field, is used to mark a BIB record in Aleph as related to an IE in Rosetta. This field includes the following subfields:

- `Yes/No indicator` – `Yes` means that this BIB record is associated with an IE in Rosetta.
- `ID field` – holds the CMS ID that is received from Rosetta.

Since a CMS record can be mapped to more than one IE, the CMS record does not hold the IE PID.

The Rosetta field is used in Aleph in the following ways:

- To create a set of BIB records that can be published by Aleph and updated in Rosetta.
- To function as a link to the Rosetta delivery of the associated IEs as part of any Aleph publishing or export routine.

To populate this field, Rosetta uses the CMS Update task, which uses a dedicated API.

CMS Management Tasks

The system employs the following two tasks 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 BIB record (in Aleph):

- **CMS Update Task** on page 13
- **Copy IE to CMS Task** on page 14

CMS Update Task

This task performs the following:

- 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 Aleph 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.
- 4 The Attach API is called to update the Rosetta field of the BIB record.

Attach API

The API uses the following parameters:

- Aleph SYS number (BIB record ID)
- CMS ID (as it is stored in Rosetta)
- Library ID
- Update flag

The API finds the matching BIB record and populates the Rosetta field:

- Yes/No indicator – Set to Yes in this indicator.
- ID field – Store the CMS ID as it is stored in Rosetta.

The API is implemented as an HTTP request, and Rosetta sends the following URL:

```
http://<base-url>?op=ros_doc&updateFlag=true&library=
<external system>&sysNumber=<recordId>&cmsId=<recordId>
```

For example:

```
http://il-aleph07:8997/
X?op=ros_doc&updateFlag=true&library=USM01&sysNumber=000028886&cmsI
d=BSB500876098
```

The baseUrl configuration is part of the parameters of the target (together with SRU parameters).

The HTTP response contains success or failure flags for every update of a CMS record or a general error description, if they occur.

In the case of a general error, the response XML (mime type - application/xml) is similar to the following:

```
<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>
```

Copy IE to CMS Task

To allow publishing from Rosetta to Aleph, create the CMS record in Rosetta before the BIB record is created in Aleph.

In this task, the CMS record is created based on the descriptive metadata stored in the IE as entered by the Producer Agent. Although duplicated (stored in the IE and as a CMS), this information acts as the CMS metadata until it is updated by the ongoing synchronization received from Aleph (if it is published from Aleph).

Configure this task as part of the Enrichment phase of the SIP Processing configurations that handle materials that are published to Aleph (for example, legal material in BSB).

The CMS ID that is generated as part of this task contains a string prefix and a number. The prefix can be unique for each institution (for example, BSB) and is defined as the parameter for this task.

NOTE:

The prefix can be different from the institution code. The Administrator can define multiple prefixes for the same institution—one for each workflow (if needed).

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 Aleph is not one of Rosetta's purposes. 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 CMS side (for example, Aleph) and use Rosetta's Delivery module for retrieving only the stream files, without the descriptive metadata.

For record deletions in Aleph, 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 the CMS source (for example, the Aleph catalog).

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 Aleph.

The detailed specification of this process in Rosetta is specified in [Ongoing Synchronization Process](#) on page 45.

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. For these CMS records, the Detach API is called to update Aleph with the information. Because these records are no longer associated with any IE, the **Yes/No indicator** of the Rosetta field is set to **No**, and the **ID field** is set to null.

Detach API

The API has the following parameters:

- CMS ID
- Library ID

This API finds the matching BIB record based on its CMS ID and deletes the Rosetta field:

- Yes/No indicator – Set this field to No.
- ID field – Set this field to null.

The API is implemented as an HTTP request, and Rosetta sends the following URL:

```
http://<base-url>?op=ros_doc_del&library=<external  
system>&cmsId=<recordId> &sysNumber= <recordId>
```

For example:

```
http://il-aleph07:8997/X?op=ros_doc_del&library=usm01&  
cmsid=123456789 &sysNumber= 12431234
```

Delivery of Multiple IEs (CMS Resolver)

Rosetta can return a list of IEs that share the same CMS ID, with the CMS ID as the query term.

The syntax of the request is as follows:


```
http://<delivery load balancer host>:<delivery load balancer  
port>/delivery/action/  
cmsResolver.do?cmsSystem=<system>&cmsRecordId=<recordId>[&dis  
playType=list&columns=<column_a,column_b> |  
&displayType=thumbnails]
```

This API is called from the resource discovery system used by Aleph (for example, Web OPAC), 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 Aleph system that is linked to Rosetta as it is written in the SRU configuration XML
- `recordId` – the CMS ID that connects the IE and the BIB record
- `displayType` - view mode to display (list or thumbnails)
- `columns` - the DC metadata fields of the IEs (list mode only)

For example:

```
http://rosetta.exlibrisgroup.com:1801/delivery/action/  
cmsResolver.do?cmsSystem=USM01&cmsRecordId=000008447-  
6&displayType=list&columns=dc:title,dc:creator
```

NOTES:

- If only one IE is returned, the request will be redirected to Delivery.
 - Content Aggregator `system` and `ID` parameters have been deprecated and should be replaced with `cmsSystem` and `cmsRecordId`, respectively.
-

Publishing from Rosetta to Aleph

The synchronization between Rosetta and Aleph is completed by the option to create BIB records in Aleph based on descriptive metadata harvested from Rosetta.

This allows institutions that catalog born digital objects in Aleph to save time and effort by entering descriptive metadata once upon ingesting the material

into Rosetta. If there is a need, the records can be managed and edited directly in Aleph, without the need to create them again.

This synchronization is supported in Rosetta by the Publishing mechanism.

Publishing in Rosetta – Overview

The Publishing module consists of the following main entities:

- **Publishing Set** – the set used as the population for publishing. For synchronization with Aleph, a logical set should be defined. This set should contain the criteria for exporting to Aleph (for example, legal documents that belong to specific Producers).
- **Publishing Profile** – the technical definitions for the Publishing module. Each profile includes the following:
 - **Converter** – the type of conversion used from the IE structure to a different structure (for example, from DC within METS to OAI-DC).
 - **Target** – the physical location to which the published IEs are written (for example, the Publishing DB table).
- **Publishing Configuration** – The combination of a publishing set and a publishing profile creates the entity called **publishing configuration**. Each configuration has a name and description. For example, publishing from Rosetta to Primo is defined in a publishing configuration and publishing from Rosetta to Aleph is defined in a different publishing configuration.

Publishing Processes

The descriptive metadata of the IEs that match the Aleph set criteria is available for Aleph to harvest. This enables the Aleph loading process to convert the DC records to MARC (MAB) and create new BIB records. An initial publishing is performed according to the publishing configurations. This is done once, converting all the IEs that are in the permanent repository and that match the Aleph set criteria. These IEs are stored in the designated location that has been defined as the target.

The following processes are designed for the ongoing synchronization between the data in the permanent repository and the data in the publishing target (the publishing table).

- **Publishing Configuration Synchronization** – This process runs the publishing according to the modified publishing configuration (if it has been modified). For example, a new criterion may have been added to the Aleph set or the publishing profile may have been removed and a new profile associated with the configuration.

- **Publishing Set Comparison** – This process is in charge of performing the publishing when there is either:
 - a change in the population of a set
 - a modification of an IE that belongs to a set that is associated with a publishing configuration.

These two maintenance processes are batch processes that run every night so as not to overload the system during business hours.

Manual Publishing Activities

Rosetta allows users (System Administrators) to perform the following:

- Create/modify the Aleph set so that it can be associated with a publishing configuration.
- Create publishing configurations by associating the set with a publishing profile.
- Create a publishing profile by adding a converter and a target.

NOTE:

For publishing to Aleph, there is a converter called `CMSConverterPlugin` that creates an `OAI_DC` with the descriptive metadata of the IE along with the CMS ID and the `URN:NBN` identifier (if it exists). The target is the `OAIPublisherPlugin`.

- Update publishing configuration by adding or removing conditions from the set or a profile from the configuration.
- Manual synchronization – After updating a publishing configuration, the user can perform the publishing process ad-hoc, based on the new configuration. This option is available only after the configuration has been modified.

NOTE:

This action is not recommended during regular business hours since it might affect system performance.

Publishing the URN from Rosetta to Aleph

The URN is a Digital Object Identifier (DOI) that is used in German-speaking countries. The following section is relevant for Rosetta users who use the URN and its related functionality in Rosetta and Aleph. (For more details, see *URN and Xepicur Support in Rosetta* in the **Rosetta > Technical Documentation > Version 2.x** section of the Documentation Center.)

The URN that is stored in Rosetta has been generated either in Rosetta or elsewhere prior to the IE ingest. It should be stored also in Aleph in the associated BIB record. There are two scenarios for copying the URN to Aleph:

- The BIB record already exists in Aleph. The IE is created in Rosetta and only the URN needs to be updated in Aleph.
- The BIB record is created based on the IE. The Aleph loader batch process creates the BIB record based on the harvest from Rosetta. These published records include the URN. The Aleph loader process is configured to map the URN to the target field of the new BIB record.

Scenario 1: Updating the URN in Existing Aleph Records

Since the BIB record already exists in Aleph, run the `updateURNTask` task to update the BIB record in Aleph with the URN value. This task calls the Aleph API called `addURN`.

See below how the `updateURNTask` is configured in Rosetta (page 44)

The API uses the following parameters:

- `urn` – Contains the URN as stored in the IE DNX.
- `bibLib` – Contains the relevant Aleph BIB library code.
- `cmsId` – Contains the CMS ID.
- `targetField` – Contains the Aleph BIB record URN field (MARC or MAB) in a six character string — for example, `552b_a` for the MAB field `552b $$a`. The first three positions in the above example contain the field code **552**, the next two positions contain the indicators (where **b** indicates the first indicator and **_** indicates an empty second indicator), and the last position contains the subfield code (where **a** indicates subfield `$$a`).

The API finds the matching BIB record based on the CMS ID and updates the target field according to its configuration.

The API is implemented as an HTTP request, and Rosetta sends the following URL:

```
http:// <base-url>? op=ros-doc-urn&cmsid=<cms-id>&URN=<URN>&targetfield=<target-field>&biblib=<library-id>
```

For example:

```
http:// il-aleph09:8997/X?op=ros-doc-urn&cmsid=BSB765765&URN=urn:nbn:de:1111-2004033116&targetfield=99800h&biblib=usm01
```

The HTTP response contains the following flags for every attempt to update a BIB record:

- `Success` – If the URN is added properly or the URN already exists.

- **Failure** – If there is a problem with one or more parameters and the update fails.
 - During SIP processing, the SIP is directed to the TA work area.
 - If the task is performed on an IE that is already in the permanent repository, the details of the IE are recorded in the process log and an indication is sent to the UI that the task has failed.

NOTE:

Multiple URNs can be added to the same BIB record, one for each IE that is associated with this CMS ID.

Scenario 2: URN in Loaded Aleph Records

Because the creation of the BIB records are based on the harvested records from Rosetta, there is no need to run additional tasks in Rosetta to update the URN field. The Aleph loader process maps the URN that is stored in the `dcterms:URN` field and populates the target field in the BIB record.

3

Error Handling

The following errors may occur during synchronization tasks:

Table 1. Error Handling

Error Name	Description	Solution
No connection between Aleph and Rosetta.	Could be due to wrong SRU configuration or a network problem.	System Administrator should make sure that: <ul style="list-style-type: none">■ the SRU XML configuration file is correct (see Configuring Rosetta on page 31).■ the network configuration allows requests in and out of Rosetta (for example, firewall definitions).
A record was not found in Aleph.	A BIB 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 is not created.	The CMS record is not created. The task aborts.	Manual assignment – Retry once the issue is resolved. Automatic assignment – The SIP is moved to the Technical Analyst - Enrichment folder. The Technical Analyst reruns the process once the issue is resolved.
The CMS ID is not generated.	The CMS ID is not generated properly. DB error. The task aborts.	Manual assignment – Retry once the issue is resolved. Automatic assignment – The SIP is moved to the Technical Analyst - Enrichment folder. The Technical Analyst reruns the process once the issue is resolved.

Table 1. Error Handling

Error Name	Description	Solution
IE is not published to Aleph.	The publishing process fails.	Review the process automation logs and/or the events that are generated (dedicated report needs to be defined).

Error handling is different for each scenario:

- **Manual assignment** – When the BIB record is assigned manually by an Assessor/Arranger or an Editor, the assignment is done online, so any issue with connecting to Aleph or creating the CMS record displays an error message. Resolve the problem and retry.
- **Automatic assignment (Pre-ingest)** – When the IE already has a populated DNX section that has the Aleph identifier in it, the system performs the CMS Update task as part of the Enrichment phase. If there are errors (for example, a record could not be found in Aleph, a connection to Aleph could not be found, or the system could not create a CMS record), the SIP is directed to the Technical Analyst – Enrichment folder where these issues can be investigated. The IE is not moved to Move to Permanent without resolving this issue.
- **Publishing to Aleph** – The events created by the publishing process (start and end time, process end status) are generated by the Process Automation and recorded in the events table. They can be viewed in a dedicated report. (This report needs to be created; it is not one of the out-of-the-box reports.)
- The publishing history of each IE is captured in an Oracle table that stores, for each IE PID, information such as publishing configuration ID, set ID, profile ID, converter, and target. This table can be the basis for reports that display the publishing statistics. (This report needs to be created; it is not one of the out-of-the-box reports.)

4

Flow Diagrams

This section includes:

- **CMS Update Task** on page 26
- **Manual Assignment Scenario** on page 27
- **Automatic Assignment – Pre-Ingest** on page 28
- **Publishing from Rosetta to Aleph** on page 29

CMS Update Task

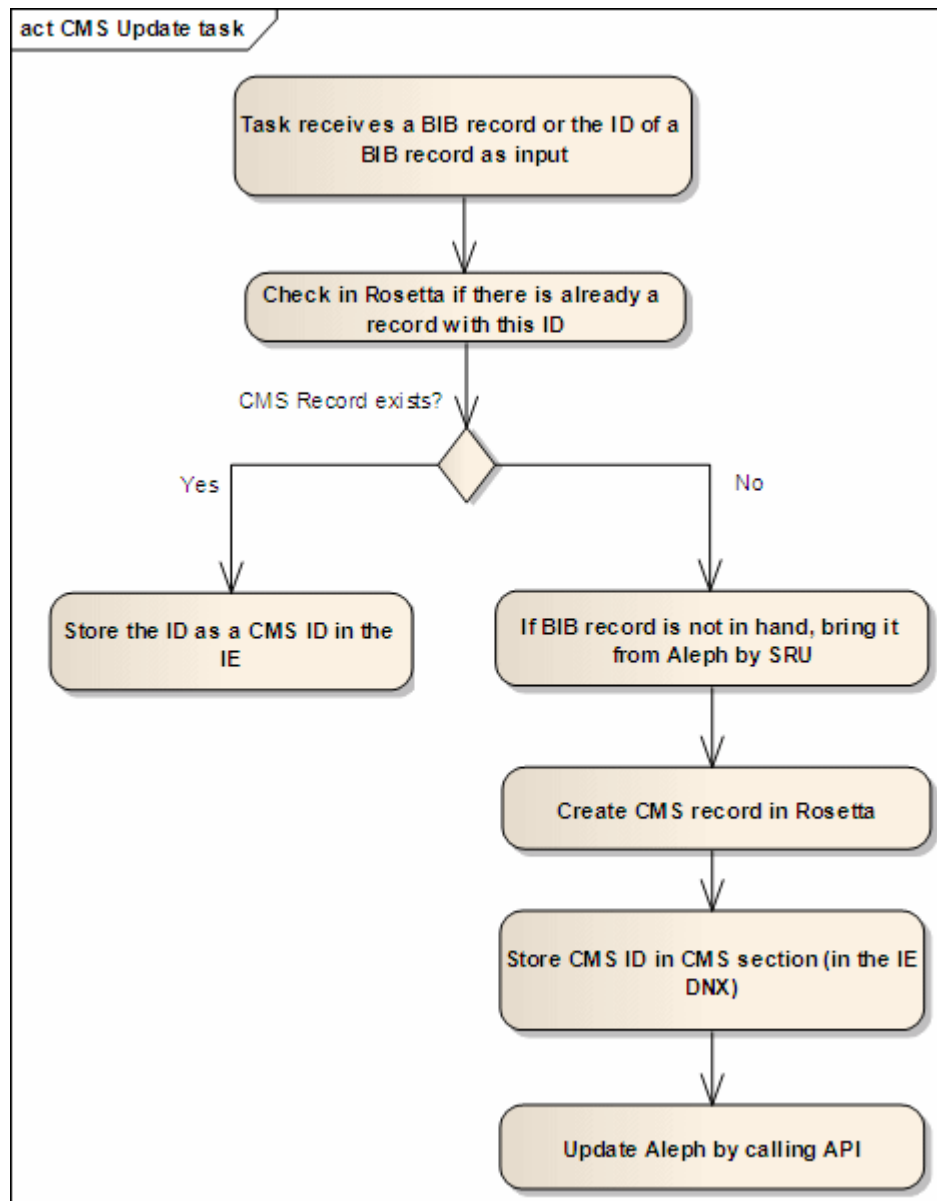


Figure 3: CMS Update Task

Manual Assignment Scenario

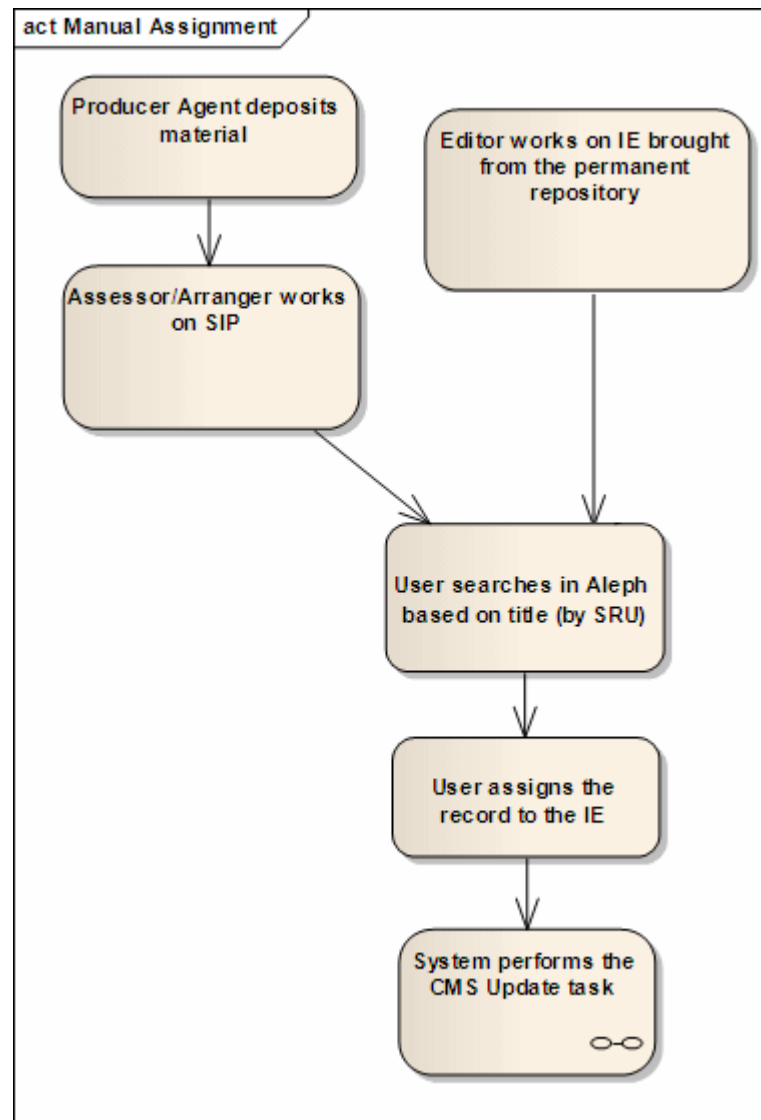


Figure 4: Manual Assignment Scenario

Automatic Assignment – Pre-Ingest

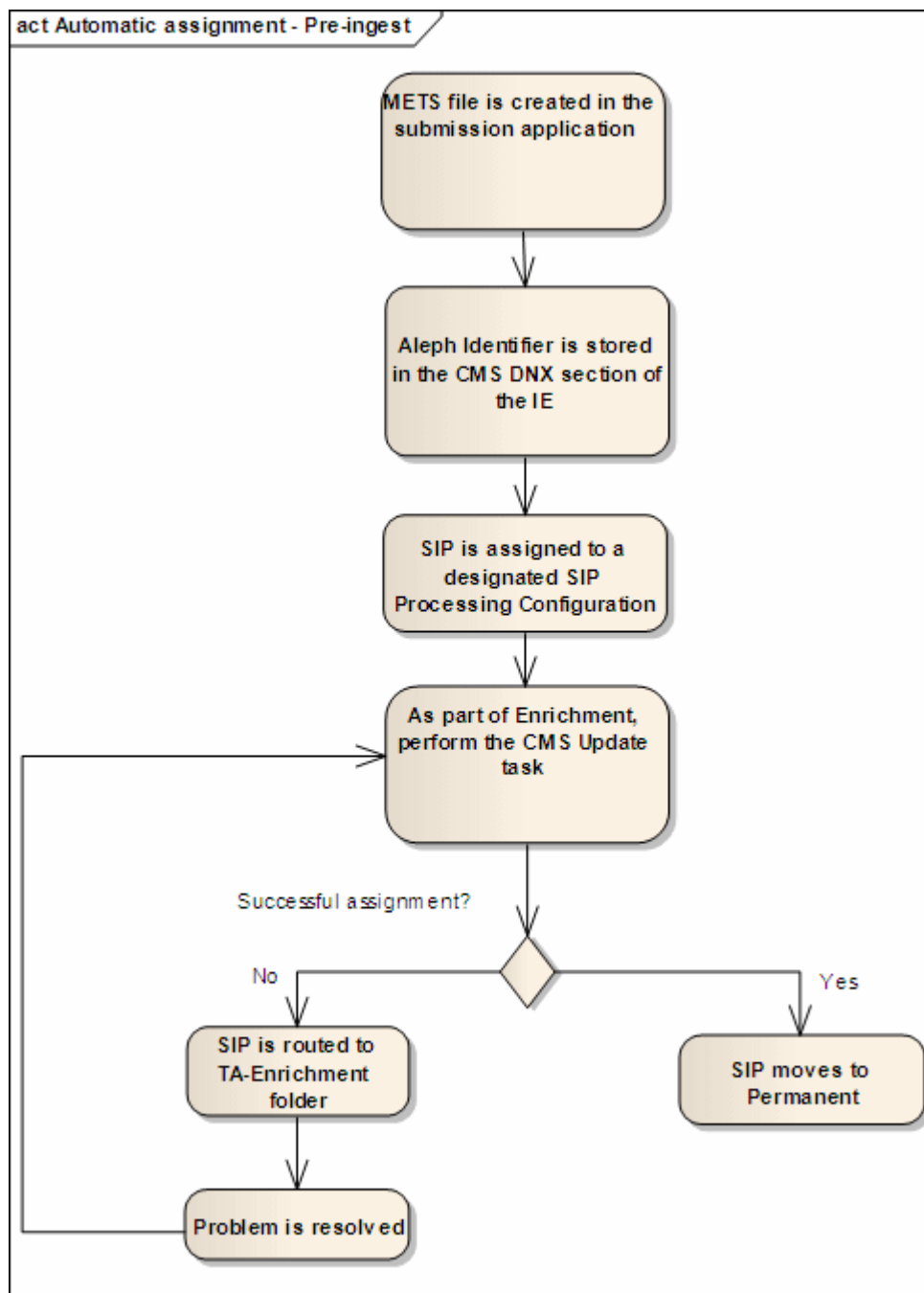


Figure 5: Automatic Assignment - Pre-Ingest

Publishing from Rosetta to Aleph

Ingest of a New IE

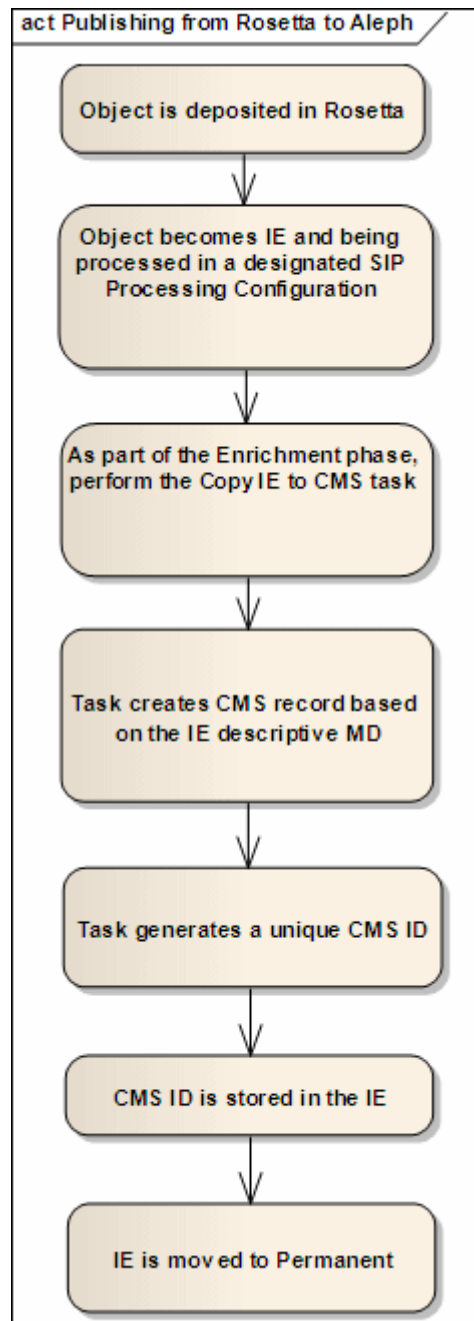


Figure 6: Ingest of a New IE

Publishing Process

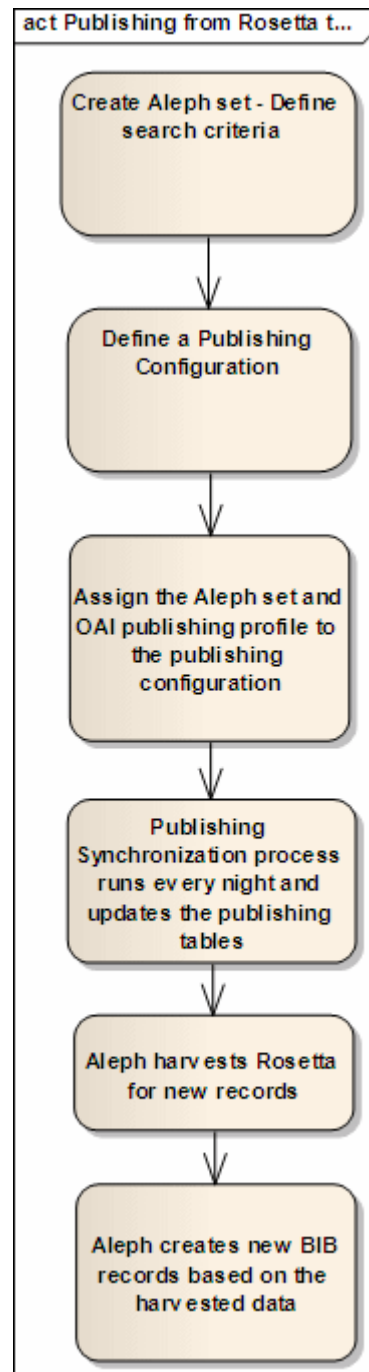


Figure 7: Publishing Process

5

Configuring Rosetta

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

This section includes:

- **SRU Definitions Configuration** on page 31
- **Configuring the CMS Update Task to Be Part of Enrichment** on page 36
- **Copying an IE to the CMS** on page 39
- **Orphan CMS Records Job** on page 43
- **Configuration of the Update URN Task** on page 43
- **Configuration of the X-Service URL** on page 44

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:
 - `repositoryName` – Set the name to the name of the Aleph library, e.g., `USM01`. The type should be set to `ALEPH`.
 - `protocol type` – Set to `SRW`.
 - `baseUrl` – The URL of the system that acts as an SRU server (CMS)
 - `version` – SRU version (currently 1.1 is supported by Rosetta)
 - `operation` – Since Rosetta is the client, `searchRetrieve` is the only option.
 - `recordSchema` – Set to `dps`
 - `indexName` – Set to `ros.id` (=ROS\$\$\$b index in Aleph)
 - `indexName2` – Set to `dc:identifier` (=MARC 001 index in Aleph)
 - `indexName3` – Set to `rec.id` (=Aleph system number index in Aleph)
 - `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.
 - `detachURL` – An Aleph API used for disassociating the BIB record in Aleph from the CMS record in Rosetta. This API clears the `ROS-ID` field and sets the `Rosetta` flag to `false`.
 - `synchUpdate` – When `synchUpdate=false`, Rosetta ignores the response received from the `updateUrl` (calls the update “asynchronic”). This allows for the attachment of CMS records and also allows for problems with the update of the external system. This parameter is optional. If missing, the default is `true`.
 - `updateURNUrl` – This parameter should be used only in German libraries, where URN is used.

The following example shows an SRU configuration for Aleph:

```
<?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="USM01" type="ALEPH">
    <protocol type="SRW">
      <parm name="baseUrl">http://il-alephrose-test:5661/usm01</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="indexName2">dc.identifier</parm>
      <parm name="indexName3">dc.ros.id</parm>
      <parm name="recordPacking">xml</parm>
      <parm name="updateUrl">http://il-alephrose-test:8991/
X?op=ros_doc&library=USM01</parm>
      <parm name="detachUrl">http://il-alephrose-test:8991/
X?op=ros_doc_del&library=USM01</parm>
      <parm name="synchUpdate">true</parm>
      <parm name="updateURNUrl">http://il-alephrose-test:8991/X?op=ros-
doc-urn&targetfield=99800h&biblib=usm01</parm>
    </protocol>
  </RepositoryName>
</ExternalResourceExplorer>
```

To get the parameters from the SRU server, call the `explain` URL. This is an example of the `explain` URL from the Aleph system:

```
<zs:explainResponse>
<zs:version>1.1</zs:version>
<zs:record>
<zs:recordSchema>http://explain.z3950.org/dtd/2.0/</
zs:recordSchema>
<zs:recordPacking>xml</zs:recordPacking>
<zs:recordData>
<explain>
<serverInfo>
<host>il-aleph07</host>
<port>9997</port>
<database>usm01</database>
</serverInfo>
<indexInfo>
<set identifier="info:srw/cql-context-set/1/cql-v1.1" name="cql"/>
<set identifier="info:srw/cql-context-set/1/dc-v1.1" name="dc"/>
<set identifier="http://zing.z3950.org/cql/bath/2.0/" name="bath"/>
<index id="4">
<title>title</title>
<map><name set="dc">dc.title</name></map>
</index>
<index id="48"><title>identifier</title>
<map><name set="dc">dc.identifier</name></map>
</index>
<index id="12">
<title>rec.id</title>
<map><name set="dc">rec.id</name></map>
</index>
</indexInfo>
</explain>
</zs:recordData>
</zs:record>
<zs:diagnostics>
<diagnostic>
<uri>info:srw/diagnostic/1/7</uri>
<message>Mandatory parameter not supplied</message>
<details>version</details>
</diagnostic>
<diagnostic>
<uri>info:srw/diagnostic/1/7</uri>
<message>Mandatory parameter not supplied</message>
<details>operation</details>
</diagnostic>
</zs:diagnostics>
</zs:explainResponse>
```

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/ 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 to Be Part of Enrichment

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.

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.

Active	Name	Description	Group	Creation Date	Modification Date	
1	RollBack Pending Entities	This task chain performs all the operations needed when removing from ...	Presentation	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
Updated By: Ribinskaite Lijana
Created By: SYSTEM
Updated On: 11/06/2010 14:09:01
Created On: 07/04/2010 17:21:23

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

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**

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

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

Figure 10: Adding the CMS Update Task

The Enrichment task chain now includes the CMS Update task.

The screenshot shows the 'Task Chain Details' page for a task chain named 'Enrichment - Digitalisate'. The page includes a breadcrumb trail: Home > Advanced Configuration > Repository > Task Chain Details. It displays metadata such as ID (4), Created By (SYSTEM), Created On (07/04/2010 17:21:23), Updated By (Ribinskaite Lijana), and Updated On (11/06/2010 14:09:01). The 'General Information' section contains fields for Name, Description, Status (Active), Status Date (07/04/2010), LogLevel (Task), and logSensitivity (All). Below this is a 'Groups' section with checkboxes for various task categories, with 'Webeditor - Permanent' and 'Enrichment' checked. At the bottom, there is a 'Task List' tab and an 'Add Task' button. The 'Task List' table shows three tasks in sequence: 1. OBJECT INDEX TASK (Object Index Task), 2. Thumbnail Task (Create Thumbnail task), and 3. Cms Update (Cms Update). Each task has dropdowns for 'Next Step On Success' and 'Next Step On Failure', and a 'Delete' button.

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

Copying an IE to the CMS

This task can be part of Enrichment, or it can be a standalone task chain that runs (as a process) on a set of IEs that are already in the permanent repository.

To configure this task:

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

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 e...	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 12: List of Task Chains

- 2 Select the **Copy IE to CMS** task chain and click **Update**.
- 3 Click the **Task Parameters** tab to update the parameters.

Task List | Task Parameters

CopyIEToCMSTask - Copy IE To CMS Task

* Persistent Identifier Generator Plug-in: CMSGenerator

* Library Code: USM01

Cancel **Save**

Figure 13: Task Parameters Tab

- 4 Enter the following parameters:
 - **Persistent Identifier Generator (CMS ID)** – This is the plug-in that generates the unique CMS ID.

NOTE:

The out-of-the-box plug-in uses **cms-** as the prefix of the generated ID. To create a different plug-in, a new plug-in must be installed (see **CMS ID Generator Plug-in** on page 41).

- **Library Code** – This parameter is used as the **External System** field in the CMS record and should match the Aleph library code (for example, USM01).

NOTE:

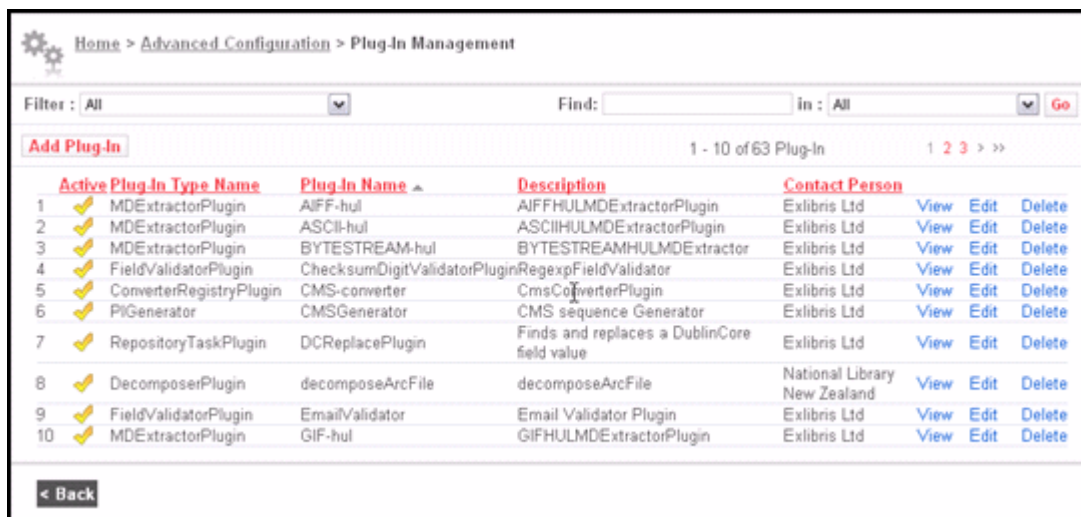
If the library code in Aleph differs from that of the Rosetta CMS records, the IEs cannot be retrieved by the CMS Resolver (that searches Rosetta by system name and CMS ID).

CMS ID Generator Plug-in

You can install a new plug-in to generate a CMS ID with a different prefix.

To install a new plug-in:

- 1 From the Administration home page, go to **Advanced Configuration > Plug-In Management**.



The screenshot shows the 'Plug-In Management' page with a table of installed plug-ins. The table has columns for 'Active', 'Plug-In Type Name', 'Plug-In Name', 'Description', and 'Contact Person'. There are 10 plug-ins listed, each with a 'View', 'Edit', and 'Delete' link. A 'Back' button is at the bottom left.

Active	Plug-In Type Name	Plug-In Name	Description	Contact Person
1	✓ MDExtractorPlugin	AIFF-hul	AIFFHULMDExtractorPlugin	Exlibris Ltd
2	✓ MDExtractorPlugin	ASCII-hul	ASCIHULMDExtractorPlugin	Exlibris Ltd
3	✓ MDExtractorPlugin	BYTESTREAM-hul	BYTESTREAMHULMDExtractor	Exlibris Ltd
4	✓ FieldValidatorPlugin	ChecksumDigitValidatorPlugin	ChecksumDigitValidator	Exlibris Ltd
5	✓ ConverterRegistryPlugin	CMS-converter	CmsConverterPlugin	Exlibris Ltd
6	✓ PiGenerator	CMSGenerator	CMS sequence Generator	Exlibris Ltd
7	✓ RepositoryTaskPlugin	DCReplacePlugin	Finds and replaces a DublinCore field value	Exlibris Ltd
8	✓ DecomposerPlugin	decomposeArcFile	decomposeArcFile	National Library New Zealand
9	✓ FieldValidatorPlugin	EmailValidator	Email Validator Plugin	Exlibris Ltd
10	✓ MDExtractorPlugin	GIF-hul	GIFHULMDExtractorPlugin	Exlibris Ltd

Figure 14: Plug-in Management

- 2 Click **Add Plug-in**. A list of plug-ins appears.

Plug-In Name	File Name	Plug-In Type Name	Plug-In Type	Install
1 Aiff-hul	AiffHUL-MDExtractor.jar	MDExtractorPlugin	java	Install
2 ASCII-hul	ASCIHUL-MDExtractor.jar	MDExtractorPlugin	java	Install
3 BYTESTREAM-hul	BYTESTREAMHUL-MDExtractor.jar	MDExtractorPlugin	java	Install
4 ChecksumDigitValidatorPlugin	ChecksumDigitValidator.jar	FieldValidatorPlugin	java	Install
5 CMS-converter	CMS-Converter.jar	ConverterRegistryPlugin	java	Install
6 CMSGenerator	CMSGenerator.jar	PIGenerator	java	Install
7 DCReplacePlugin	DCReplacePlugin.jar	RepositoryTaskPlugin	java	Install
8 decomposeArcFile	ArcDecomposerPlugin-1.0.jar	DecomposerPlugin	script	Install
9 GIF-hul	GIFHUL-MDExtractor.jar	MDExtractorPlugin	java	Install
10 HTML-hul	HTMLHUL-MDExtractor.jar	MDExtractorPlugin	java	Install
11 IETOA-Converter	IETOA-Converter.jar	ConverterRegistryPlugin	java	Install
12 JPEG-hul	JPEGHUL-MDExtractor.jar	MDExtractorPlugin	java	Install
13 JPEG2000-hul	JPEG2000HUL-MDExtractor.jar	MDExtractorPlugin	java	Install
14 MP3toWaveMigrationTool	MP3toWaveMigrationTool.jar	MigrationToolPlugin	script	Install
15 NFS-Publisher	NFS-Publisher.jar	PublisherRegistryPlugin	java	Install
16 NLB_PID_Insert_Plugin	NLBPIDInsert-Plugin.jar	RepositoryTaskPlugin	java	Install
17 NLB_PID_Plugin	NLBPID-Plugin.jar	RepositoryTaskPlugin	java	Install

Figure 15: List of Plug-ins

- 3 Select the **URNFixedLengthGenerator** plug-in and click **Install**. The form for defining a new CMS ID generator opens.

Plug-In Information			
Plug-In Id	-	Plug-In Name	URNFixedLengthGenerator
Plug-In Type Name	PIGenerator	Implementation Type	java
Implementation Name	com.exlibris.dps.repository.pl...	Interface	-
Owner	-	Material Type	DIGITAL
Family Type	TASK	Description	URNFixedLengthGenerator
Resource Type	-	Module	Repository
Status	-	Public API	false

Contact Information			
Contact Type	admin	Email address	yaarac@exlibris.co.il
Last Name	Ltd	First Name	Exlibris
Address 1	Agodat Asport2 Building 9	City	Jerusalem
Telephone 1	-		

Plug-In Parameters	
* Prefix	urn
* Type	URN

Back Cancel Install

Figure 16: Plug-in Information

- 4 Assign a unique name and description to the plug-in and edit the following parameters:
 - **Prefix** – This is the prefix of the CMS ID. For example, if the numbers are generated by a database sequence, the IDs that are generated by this plug-in could be **BSB-0**, **BSB-1**, **BSB-2**, and so forth.
 - **Type** – This parameter should be set to **CMS**.
- 5 Click **Install**. The plug-in is available for selection as a value of the **Persistent Identifier Generator** in Step 4 in [Copying an IE to the CMS](#) on page 39.

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.

Configuration of the Update URN Task

The UpdateURNTask task can be performed on a set of IEs as a stand-alone task chain, or it can be part of the Enrichment task chain, as shown below.

The screenshot shows the 'Task Chain Details' configuration page in Rosetta. The breadcrumb trail is 'Home > Advanced Configuration > Repository > Task Chain Details'. The task chain is named 'Enrichment - Minimal' and is currently 'Active'. The status date is 26/04/2011. The log level is set to 'Task' and log sensitivity is 'All'. Under the 'Groups' section, several tasks are listed with checkboxes: Validation Stack, Workbench, Webeditor - Staging, Preservation, Move To Permanent, Maintenance, Webeditor - Permanent, Metadata Validation, and Enrichment. The 'Task List' tab is selected, showing a table of tasks in the chain:

Set order	Name	Description	Next Step On Failure	
1	Cms Update	Cms Update		Delete
2	PIGeneratorGenericTask	Persistent Identifier Generator Task		Delete
3	UpdateURNTask	Update URN Task		Delete
4	OBJECT INDEX TASK	Object Index Task		Delete

At the bottom of the page are 'Cancel' and 'Save' buttons.

Figure 17: Task Chain Details

NOTE:

There are no parameters for this task.

Configuration of the X-Service URL

Add the bold text to the External Resources configuration file, as shown below:

```
<?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="USM01" type="ALEPH">
    <protocol type="SRW">
      <parm name="baseUrl">http://193.174.96.246:5661/bvbros</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="indexName2">dc.identifier</parm>
      <parm name="indexName3">dc.ros.id</parm>
      <parm name="recordPacking">xml</parm>
      <parm name="updateUrl">http://193.174.96.246:8991/
X?op=ros_doc&library=bvb01</parm>
      <parm name="detachUrl">http://193.174.96.246:8991/
X?op=ros_doc_del&library=bvb01</parm>
      <parm name="updateURNUrl">http://il-aleph09:8991/X?op=ros-doc-
urn&targetfield=99800h&biblib=usm01</parm>
    </protocol>
  </RepositoryName>
</ExternalResourceExplorer>
```

Figure 18: External Resources Configuration File

The above file can be accessed from the Advanced Configuration > General > Configuration Files page.

6

Ongoing Synchronization Process

This section includes:

- **Introduction** on page 45
- **Output File Naming Convention** on page 46
- **Export File Content** on page 47
- **Error Handling** on page 48

Introduction

This section describes the process used for supporting ongoing synchronization.

On the Aleph side, the Publishing mechanism is used for exporting records to Rosetta, in the Rosetta DC format, using a `*.tar.gz` 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 to which both Aleph and Rosetta have access.

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

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.FileAdapterForOA
I</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="verbose" value="false"></
adapter_param>
  </synch_job>
-->
</metadata_job_conf>
```

Output File Naming Convention

The list of all published CMS records are exported to a *.tar.gz 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>.<set
name>.<yyyyMMdd>.<hhmmss>.<sequence>.<tar.gz
```

Where the repositoryCode is the name of the CMS (for example, Aleph) 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 the following name:

```
aleph.ROSETTA_PUB.20101230.085202.1.tar.gz
A larger export (broken into three files) might have the
following names:
aleph.ROSETTA_PUB.20101230.085202.1.tar.gz
aleph.ROSETTA_PUB.20101230.085202.2.tar.gz
```

Files that are not compliant with the above convention are not loaded and are 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:

```
<?xml version="1.0" encoding="UTF-8"?>
<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 19: 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>:<Aleph System
number>:<MARC 001>:<CMS ID>:</identifier>
```

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<header>
  <identifier>DPS:il-aleph07:USM01:0000000001:0000000001</identifier>
  <timestamp>2010-12-16T05:29:05Z</timestamp>
  <setSpec>ROSETTA</setSpec>
</header>
```

Figure 20: Format of the Identifier

Note that the repository code and the host name must be identical to the way in which they are defined in the SRU configuration XML that is used for defining the connectivity between Rosetta and Aleph (see [Configuring Rosetta](#) on page 31).

Note that if the record is deleted, the status attribute is as follows:

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

In the case of an updated document, the DC part of the CMS record is overwritten by the DC section below:

```
- <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/dcterms/"
  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 21: 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 occurred during the process.

7

Configuring Aleph

This section includes:

- **Overview of Integration Workflow** on page 49
- **Publishing Aleph BIB Records to Rosetta** on page 50
- **Aleph SRU Server Setup** on page 52
- **Rosetta CMS Resolver URL** on page 52
- **Aleph SRU Server 001 Search Setup** on page 53
- **Aleph Z39.5 Server Setup** on page 54
- **Expand Routines** on page 54
- **Defining Translation in Aleph of DC XML** on page 54
- **Defining a Rosetta Link Display in Aleph Web OPAC** on page 55
- **Detaching the API Index** on page 55
- **Harvesting Records from Rosetta** on page 56

Overview of Integration Workflow

The following is a description of the Rosetta-Aleph integration workflow from an Aleph perspective:

- 1 Rosetta sends Aleph an SRU request (“explain” request) to get the list of indexes supported by the Aleph SRU server. Aleph returns the active indexes.
- 2 Rosetta sends Aleph an SRU search request according to one of the SRU supported indexes. For example:

```
http://il-aleph07:5667/usm01?version=1.1&operation=searchRetrieve&query=dc.title=Memories&maximumRecords=10&recordSchema=dps
```

This returns a list of Aleph records in DC format.

- 3 Rosetta sends Aleph an SRU search request based on the `cms_id` and gets a record from Aleph in the DC format. For example:

```
http://il-aleph07:5667/usm01?version=1.1&operation=
searchRetrieve&query=dc.identifier=123456789&maximumRecords=1
0&recordSchema=dps
```

The returned record includes the `cms_id` (BIB control number) in the `<dc:identifier>` XML tag. Aleph uses the `expand_doc_ros_id` program to create it.

- 4 Rosetta sends Aleph a message when a record has been imported (attached) or removed (detached). This is done using the X-service. The Aleph record that is linked to Rosetta is populated with a `ROS$aY` tag that indicates the link to Rosetta.
- 5 Aleph exports records to Rosetta in a `.tar` file using the DC format. The Aleph publishing process defines a base set that includes records linked to Rosetta objects (or all records with the same ROS index). The records are published in a DC XML.
- 6 Any kind of discovery tool will be able to display a link to the CMS Resolver in Rosetta for records that are exported to Rosetta. The Aleph `expand` program (`expand_doc_link_to_ros`) works on records with `ROS$aY` tags, creates a link to Rosetta (using the Rosetta CMS Resolver URL address, which is defined as an Aleph environment variable), and includes the `cms_id`.

Publishing Aleph BIB Records to Rosetta

Using the Aleph publishing mechanism, BIB records with the `ROS$aY` field are published to Rosetta in a `.tar` file. This is done by defining a base set that includes records belonging to Rosetta (a base set for all records with the same ROS index).

The following is a sample setup for defining a base (ROSBASE) for BIB records with `ROS$a`. `ROS$a` is indexed to a dedicated word index (for example, `wrs`).

Base configuration example:

Add in `./alephe/tab/tab_base.lng`:

```
!           1           2           3           4           5           6           7           8           9
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!>
ROSBASE      Rosetta publishing    USM01      USM01 Y wrs=Y
```

Index configuration example:

Add in `./<bib library>/tab/tab11_word:`

!	1	2	3	4	5	6	7	8	9	10	11	12
!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!	-!!!!!!
!!!!!!												
ROS				a				01		WRS		

To activate this configuration, `manage_01` should be run in the BIB library and the `UE_01` daemon should be restarted.

To publish Aleph records (in DC XML format) that are linked to Rosetta, you must define a publishing set in the `./<bib library>/tab/tab_publish` file.

The publishing set should point to a base set that includes all Aleph records linked to Rosetta (a base set for all records with the `ROS$aY` ROS index).

Publishing set configuration example:

In the following example:

- **Column 1** – The publishing set is named ROSETTA.
- **Column 2** – The set is defined for documents contained in the logical base ROSBASE. This base has been defined in advanced in `./alephe/tab/tab_base.<lng>` as the logical base set of all documents with the same ROS index.
- **Column 4** – Contains the ROS expand routine set in `tab_expand` for `expand_doc_ros_id`.
- **Column 5** – Contains the `ROS_OAI_DC_XML` DC XML format.

Add in `./<bib library>/tab/tab_publish:`

!	1	2	3	4	5
!!!!!!	!!!!!!	!!!!!!	!!!!!!	!!!!!!	!!!!!!
ROSETTA	ROSBASE	N	ROS	ROS_OAI_DC_XML	

NOTE:

The 4th column in `./<bib library>/tab/tab_publish` determines which fix and expand routines will be activated according the configuration in the `./<bib library>/tab/tab_expand` and `./<bib library>/tab/tab_fix` tables.

The publishing mechanism is activated by the initial processing of a set of documents (batch service: `publish-04`) and the exporting of `.tar` files for the published set (batch service: `publish-06`).

Aleph SRU Server Setup

To allow a search and retrieve in Aleph through the SRU server, add the explain section to `./alephe/tab/sru_server/config.xml`.

- In the `<host>` tag, enter the Aleph IP machine.
- In the `<port>` tag, enter the Z39.5 server port number.
- In the `<database>` tag, enter the Aleph BIB library code.

For example:

```
<explain xmlns="http://explain.z3950.org/dtd/2.0/">
  <serverInfo>
    <host>il-aleph07</host>
    <port>9991</port>
    <database>usm01</database>
  </serverInfo>
  <indexInfo>
    <set identifier="info:srw/cql-context-set/1/cql-v1.1" name="cql"/>
    <set identifier="info:srw/cql-context-set/1/dc-v1.1" name="dc"/>
    <set identifier="http://zing.z3950.org/cql/bath/2.0/" name="bath"/>
    <index id="4">
      <title>title</title>
      <map><name set="dc">dc.title</name></map>
    </index>
    <index id="12">
      <title>rec.id</title>
      <map><name set="dc">rec.id</name></map>
    </index>
    <index id="122">
      <title>ros.id</title>
      <map><name set="dc">ros.id</name></map>
    </index>
  </indexInfo>
</explain>
```

Rosetta CMS Resolver URL

In the `./exlibris/aleph/a20_1/alephe/aleph_start.private` file, add a line for the Rosetta CMS Resolver URL address.

For example:

```
http://il-dtldev08c.corp.exlibrisgroup.com:1801/delivery/action/
cmsResolver.do?
```

Aleph SRU Server 001 Search Setup

To configure the SRU server to support searches on the BIB 001 field (system number):

- 1 In the index translation section in `./alephe/tab/sru_server/pqf.properties`, enter `index.dc.identifier:1=48`.
- 2 In the Word translations section in `./alephe/tab/z39_server/z39_server_USM01.conf`, enter the sample set, `rsn`, as a direct index to search the 001 field:

```
word      rsn      48
```

To enable the search on 001 field in Aleph:

- ```
1 In ./<BIB library>/tab/tab00.lng, add:
```

```

! 2 3 4 5 6 7 8 9 10 11
!-!!!!!!-!!!!!!-!-!!-!!-!!-!---!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!
H RSN IND 21 00 00 Rosetta System Number

```

- ```
2 In ./<BIB library>/tab/tab11_ind, add:
```

```

1      2      3      4      5      6      7 8
!!!!-!!!!-!-!!!!!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!-!-!
001##                      RSN

```

- 3 Run `manage-05` in update mode.
- 4 Restart `UE_01` using `util/e/1`.
- 5 Restart the Z39 server using `util/w/3/4`.
- 6 Restart the SRU server using `util/w/3/7/8`.
- 7 Restart the PC server using `util/w/3/3`.
- 8 Restart the WEB server using `util/w/3/1`.

Aleph Z39.5 Server Setup

To define the SRU server connection, add lines similar to the following to the `z39_server_<BIB>.conf` file (for example, `./alephe/tab/z39_server/z39_server_USM01.conf`):

```
#####
#Present Settings
#####
out-record-syntax XML
out-record-expand ROS
```

In `out-record-expand`, define the `expand` routine you are about to implement in `tab_expand` for `expand_doc_ros_id` (ROS in the above example).

Expand Routines

Define the following two routines in the `./<bib_library>/tab/tab_expand` file:

- **expand_doc_ros_id** – for generating the `cms_id` (Aleph BIB control number). Make sure the routine name (set in the first column of `tab_expand--ROS` in the example below) is used in the `out-record-expand` section of `Z39_server_<BIB>.conf` as well. This expands the BIB record with the field `RST`.
- **expand_doc_link_to_ros** – to display, in Aleph Web OPAC, a link to the CMS Resolver in Rosetta for records that are exported to Rosetta. \

```
Sample: ./usm01/tab/tab_expand
!      1                      2                      3
!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
ROS                expand_doc_ros_id
WEB-FULL           expand doc link to ros
```

Defining Translation in Aleph of DC XML

Add the following lines to the `./<bib_library>/tab/tab 12` file, which defines the translation of Aleph fields and sub-fields to unqualified Dublin Core XML elements:

```

1          2          3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
RST a      identifier

```

Note that:

- Column 1 should always be set to `RST`.
- Column 2 should always be set to `a`.
- Column 3 should always be set to `identifier`.

Defining a Rosetta Link Display in Aleph Web OPAC

To set a link to Rosetta objects in the Aleph Web OPAC, add the following lines to the `./<bib_library>/tab/edit_doc_999.eng` file:

```

!      1      2      3      4      5 6          7          8 9 10 11 12
!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!>
##  RSTLK      D L Rosetta Link      Y S      E

```

NOTE:

Column 2 should always be set to RSTLK.

Detaching the API Index

The following configuration should be set in order to detach an Aleph Bibliographic document from Rosetta using the detach API.

- 1** Add the following in `./<BIB library>/tab/tab00.lng`:

1	2	3	4	5	6	7	8	9	10	11
!	-	!	!	!	-	!	!	-	!	!
H	CMSID	IND			21	00		00		Rosetta System Number

- 2** In `./<BIB library>/tab/tab11 ind, add:`

! 1 2 3 4 5 6 7 8
 !!!!!-!!!!-!-!!!!!!-!!!!-!!!!!!!!!!!!!!!!!!!!!!-!-!
 ROS CMSID b

The detach API removes the ROS field from the Bibliographic document using an Aleph index on the ROS\$\$b data.

Harvesting Records from Rosetta

Aleph uses the standard OAI harvester (ue_13) to harvest records published by Rosetta. The transferred data is stored in the `$alephe_scratch` directory. The harvested records are in the DC format.

The harvester starts the `p_rosetta01` service which performs the following steps:

- 1 Converts DC to Aleph-native metadata format—for example, MARC or MAB.
- 2 Sets the Rosetta indicator to **yes** (`ROS__$aY`).
- 3 Creates new BIB records.
- 4 Loads the new BIB records into Aleph.

The service loads only records with a new CMS ID in field `ROS__$b`. This means that records are rejected if they do not contain a CMS ID or if the ID already exists in Aleph.

The files in `$alephe_scratch` have the following naming convention:

```
ue_13_<date>.<seq.number>_<setname>
<date> - YYYYMMDD
<seq.number> - 5-digit sequential number
<setname> - Name of the [Sets] section in ue_13 configuration
```

To run the harvesting and loading, the following needs to be configured:

- **Aleph OAI Harvester** on page 57
- **Aleph rosetta-01 Loader** on page 60
- **Format Conversion** on page 61

Aleph OAI Harvester

The table `./<BIB library>/tab/tab_ue13.conf` is used to configure the Aleph OAI harvester (ue_13). Define the following parameters for this process:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[Main]
BASE-URL = http://server:port/oaiprotider/request
!Base URL of the (Rosetta) OAI provider.

DELETE-XML-FILES = N
!Options (Y/N) - Delete XML files from $alephe_scratch after
processing the records.

DELETE-LOG-FILES = N
!Options (Y/N) - Delete log files from $alephe_scratch after
processing the records.

METADATA-PREFIX = oai_dc
!Format which has been provided from Rosetta, "oai_dc" is
supported only.

RECALL-PERIOD = 24h
!Value for the time period for every harvesting step (in minutes
unless otherwise specified - h = hours). It is a time range for
each
OAI request - the range between "from" and "until" parameters.
This parameter can contain two digits.
This means that the maximum amount of minutes can be 99.
For value larger than 99 minutes - specify "h" (hours).
```

```

SERVICE-NAME = oairep.pl
!Perl script which is used by ue_13.

START-DATE = 2011-01-01-T00:00:00Z
!Time stamp (in UTC) for provided data. The records must have
been provided after this date.
!Syntax: YYYY-MM-DD-THH:MM:SSZ This parameter is only
relevant for the first run of the ue_13 and from then on will
be run according to the ue-13 last date and time.

TIMEOUT = 900
!Timeout value for ue_13 OAI harvester (in seconds).

WAIT-TIME = 216000
!Waiting time before checking for new records (in seconds).
For instance, 216000 seconds = 24 hours.

[Debug]
VERBOSE-MSG = 2
!The amount of information to be printed in the log files (1,
2)

[Sets]
ROSALEPH1 = Set
!Name of the Rosetta OAI set.
ROSALEPH1-RUN-01 = csh -f $aleph_proc/p_rosetta_01
MAB01,<filename>,<setname>,,,DCMAB,,,
!Example of command to start service rosetta-01.

```

Harvester Timestamps

Timestamps are counters in Oracle table Z52. They can be managed using the libraries UTIL G/2 menu:

```

G. Tables for Defining Database Structure
-----
0. Exit Procedure
1. Database Tables (Tab Files)
2. Display/Update Library Parameters
...

```

For example:

	Sequence Name	Value	Suppress	Type	Prefix
	-----	-----	-----	-----	-----
19.	ue13-last-date	20110207			
20.	ue13-last-time	135008			
21.	ue13-run-date	20110207			
22.	ue13-run-no	9			

The following table describes the timestamps:

Table 2. Harvester Timestamps

Timestamp	Description
ue13-last-date	End date of last harvesting – YYYYMMDD
ue13-last-time	End time of last harvesting – HHMMSS
ue13-run-date	Date of last start of ue_13
ue13-run-no	Number of ue_13 starts

The last date and last time are used as the parameters for the next harvesting. If a re-harvest needs to start in the past, ue13-last-date and ue13-last-time can be set to the appropriate values. ue_13 must be restarted to activate any new parameters.

Starting and Stopping ue_13

ue_13 is a background daemon running in the Aleph library. It is managed using the libraries UTIL E menu:

```
E. Monitor Background Jobs
-----
0. Exit Procedure
1. Start Update Doc Index
2. Stop Update Doc Index
5. Store/Restore History of Handled Doc Index
6. Start Request Handling (ADM Library Only)
7. Stop Request Handling (ADM Library Only)
8. Start Update BIB Acc from Aut (Acc + Doc)
10. Stop Update BIB Acc from Aut
11. Start Messaging Update (z105 Messaging Library
only)
12. Stop Messaging Update (z105 Messaging Library only)
13. Start OAI Harvester (BIB Library Only)
14. Stop OAI Harvester (BIB Library Only)
...
```

Tracking ue_13 and rosetta-01

The ue_13 harvester log file is located in the \$data_scratch directory of the Aleph library. It tracks both the harvesting as well as the processing of p_rosetta_01. The log file name is in the following syntax:

```
run_e_13.<timestamp>
```

Aleph rosetta-01 Loader

The following is the structure of the command to run rosetta-01:

```

csh -f $aleph_proc/p_rosetta_01 <BIB
library>,<filename>,<setname>,<report file name>,<cms
index>,<fix procedure>,<cataloger name>,<cataloger level>

```

For example:

```

csh -f $aleph_proc/p_rosetta_01
MAB01,<filename>,<setname>,,,DCMAB,,,

```

The following table describes the parameters of Aleph rosetta-01 loader:

Table 3. Aleph rosetta-01 Loader

Parameter	Description
<filename>	The placeholder for the name of the input file that contains the metadata harvested from Rosetta. The file must be located under \$alephe_scratch..
<setname>	The placeholder for the name of the Rosetta OAI set (currently without functional meaning).
Report File Name	The name of the report file that is created during the process, with information about the BIB records that have been created. Default: <input file>.doc_log
CMS index	Index name of the Aleph CMS ID (content of ROS__\$b). Default index name: CMSID
Fix procedure	Reference to the fix procedure defined in ./<BIB library>/tab/tab_fix, that is used for the conversion of the input data to Aleph format (MARC, MAB, UNIMARC, and so forth). This includes the conversion of Rosetta CMS ID to Aleph field ROS__\$b.
Cataloger Name	If the user name of the cataloger is entered in this field, a CAT field containing the cataloger name is added to all created records.
Cataloger Level	This field is used to determine the cataloger level that is recorded in the CAT field added to a record that has been added.

Format Conversion

The conversion of DC elements to Aleph fields is done in two steps. First, the DC metadata is converted to Aleph sequential format. Second, Aleph sequential format is converted to Aleph metadata format (MARC, MAB, UNIMARC, and so forth).

The table `./<BIB library>/tab/tab_dc` is used to configure the first step. The following is the structure of the table:

```
Col. 1: Internal field code
Col. 2: Dublin Core Element
!1      2
!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!>
001    dc:title
002    dc:creator
...
008    dc:identifier
...
```

The internal field codes are referenced in the related mapping table (for example, `tab_fix_dc2mab`) which is used for the next step of the conversion from DC to Aleph metadata format.

The table `./<BIB library>/tab/tab_fix` defines the Aleph fix routine and the related mapping table.

The following is an example of the table structure:

```
!Col. 1: Fix routine name
!DCUSM - Name of the fix routine to convert DC-XML to Aleph-
MARC
!DCMAB - Name of the fix routine to convert DC-XML to Aleph-
MAB
!
!Col. 2: Fix program name
!fix_doc_convit is the program used.
!
!Col. 3: Program arguments
!FILE= Mapping table name

! 1      2                                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
DCUSM fix_doc_convit FILE=tab_fix_dc2usm
DCMAB fix_doc_convit FILE=tab_fix_dc2mab
```

NOTE:

In addition to `fix_doc_convit`, all programs that are defined for fix routine INS are used when loading new Aleph records from Rosetta.

The mapping tables—for example, `./<BIB library>/tab/tab_fix_dc2mab-` contain rules for the conversion of internal field code into Aleph categories (conversion step 2). This description is valid for any configuration table that may be used together with the `fix_doc_convit` program.

The following is an example of the table structure:

```
Col. 1: Source Tag and Indicator
Col. 2: Source Subfield
Col. 3: Target Tag and Indicator
Col. 4: Target Subfield
Col. 5: Program Name with Arguments
! 1 2 3 4 5
!!!!-!-!!!!-!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
##### LDR const_field
"00000nM2.01200024|||||h",ONCE-IDN=01
##### 030 const_field "zz|||||||17",ONCE-IDN=02
##### 050 const_field "|||||||a|||||",ONCE-IDN=03
##### FMT const_field "MH",ONCE-IDN=04
001## a 331 a
...
008## a 552b a chkit_contains CHK-SRC=b,CHK-
TXT="dcterms:URN"
008## a ROS b chkit_contains CHK-SRC=b,CHK-
TXT="rosetta:CMS"
...
```