Rosetta Integrations with External Systems Guide

Version 2.2.1

CONFIDENTIAL INFORMATION

DISCLAIMER

TRADEMARKS

Document released: November 2011

Web address: http://www.exlibrisgroup.com

Table of Contents

1

Overview

Rosetta integrates with external systems (such as Aleph, Primo, and Voyager) via standard protocols (such as SRU, OAI-PMH) and specifically developed APIs.

This document reviews the integrations, the protocols used, and the guidelines for adding new integrations.

The integrations are in the following areas:

- **CMS** – Content Management Systems (such as Aleph and Voyager) can be integrated with Rosetta in such a way as to support the synchronization of metadata between IEs in Rosetta and matching IEs (or "documents") in the external systems. In this way, IE(s) in Rosetta can be associated with richer and better maintained descriptive metadata that can be retrieved during delivery and displayed as a whole to the end user. The CMS integration uses the SRU protocol and dedicated APIs that were developed for each integration separately.
- **Publishing** – Resource Discovery applications such as Primo can harvest Rosetta for information about IEs by using the OAI-PMH protocol. Other system can harvest Rosetta with no further development.
- **DOI** - A digital object identifier (DOI) is a character string (a digital identifier) that uniquely identifies an object, such as an electronic document. Rosetta currently integrates with three different systems (Handle, URN:NBN, and NLB PID) and provides a generic mechanism that allows other types of DOI to be generated and published in Rosetta.
- SDK – Rosetta Software Developer's Kit (SDK) allows users to integrate Rosetta with applications of two kinds:
  - Submission Applications – For generating METS files and depositing IEs into Rosetta.
  - Viewers – For integrating with the Rosetta Delivery module and allowing the IEs to be viewed via different tools than what is available in Rosetta default set-up.

For information about the available SDK, see the documentation in Ex Libris EL Commons.

2

CMS Integration

This section includes:

Introduction

Rosetta's synchronization with Aleph, Voyager, and Tapuhi supports the following scenarios:

- Manual search in CMS by record title (using the SRU protocol) and assignment of a CMS record to an IE in Rosetta.
- Automatic pre-ingest assignment of a CMS record ID to an IE.
- Delivery by an SRU request (search in Rosetta by CMS ID) and receiving 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 in Rosetta can be updated via updates that arrive from CMS. These records are expected to be in OAI DC format.
- The record in CMS has an indication that it is associated with IEs in Rosetta.

Management of CMS ID

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

The CMS ID can be the record identifier of the CMS record (such as the system number in Aleph and Voyager) or a different identifier that is generated in CMS (such as MARC 001 in Aleph).

Rosetta supports a work flow where the CMS ID is generated in Rosetta and published to CMS (Aleph) via OAI.

NOTE:
With Voyager integrations, only the system number is used as the CMS ID. In an integration with Aleph, the ROS field is used to hold the CMS ID. It can contain either the system number of the same record, the MARC 001 field of the same record, or a unique CMS number that was generated by Rosetta.

Search and Retrieve Records from CMS 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 CMS 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 CMS 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.

CMS Implementations

Out of the box, Rosetta integrates with Aleph, Voyager, and Tapuhi. Before working with these systems, you must configure your system according to the

instructions in the *Rosetta - Aleph Synchronization* or *Rosetta - Voyager Synchronization* guides. (Integration with Tapuhi is the same as the integration with Voyager.)

These guides are available in the Ex Libris Documentation Center.

Future Integrations

Institutions that would like to integrate Rosetta with other systems could implement them as a CMS integration, based on SRU.

Integration of these systems will require analysis and probably development on both sides.

3

Publishing Integration

This section includes:

Introduction

The Publishing module allows every part of the IE's metadata to become available to external systems for harvesting.

The publishing module consists of the following main entities:

- **Publishing Set** – the set that is used as the population for publishing.

  For example, a logical set is defined for synchronization with Aleph. This set contains the criteria for exporting to Aleph (such as 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

An initial process of 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 publishing set criteria. The published data is stored in a 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 (OAI table or NFS folders).

- **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.

The above maintenance processes are batch processes that run every night to prevent overloading the system during business hours.

Manual Publishing Activities

Rosetta allows users (Data Managers) to perform the following:

- Create/modify the publishing 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.
- Update publishing configuration by adding or removing conditions from the set or a profile from the configuration.
- Perform manual synchronizations. After updating a publishing configuration, users can perform the publishing process ad-hoc, based on the new configuration. This option is available only after the configuration has been modified.

Publishing Events

Events are captured by the system and written to the Events table. The events created by the publishing process (start and end time, process end status) are generated by the process automation and can be viewed in a dedicated report.

NOTE:
This report is not one of the out-of-the-box reports and needs to be created.

Publishing History

The publishing history of each IE is captured in an Oracle table that stores information such as the publishing configuration ID, the set ID, the profile ID, and the converter and target for each IE PID.

This table can be the basis for reports that show publishing statistics.

NOTE:
This report is not one of the out-of-the-box reports and needs to be created.

Publishing Integration - Implementation

The out-of-the-box implementation of Rosetta integration (Rosetta - Primo) uses the OAI-PMH protocol for harvesting published data from Rosetta.

For more details, see the *Rosetta – Primo Integration Guide* in the Ex Libris Documentation Center.

Future integrations can be implemented with any system that uses OAI-PMH.

Publishing Format

Users of Rosetta can configure and use XSL files for converting the METS XML to any kind of XML for holding relevant parts of the IE metadata.

The XSL can be a simple conversion from METS to DC (Dublin Core) or a conversion of some DC fields and some DNX fields.

Adding new XSL files does not require any upgrade of Rosetta. The XSL files can be found and added to the following folder in each of Rosetta's installations:

```
/exlibris/shared/operational/xsl/publishing
```

Full Text Publishing

In order to support full text search in Primo for PDF files that are stored in Rosetta, the published metadata can include the physical location of the files in the permanent repository. This allows Primo to copy the files themselves and convert them to text, so their entire context will be searchable by the end-users.

NOTE:
This implementation requires that Primo server to have access to the file system that Rosetta uses as the permanent repository.

The `DC2OaiWithFilePath.xsl` file, which includes the full text support, is located in the following folder:

```
/exlibris/shared/operational/xsl/publishing
```

The following example shows an OAI DC record that includes the file location in the DC field **dcterms:fullTextPath**:

Figure 1: OAI DC Record Example

Publishing Target

The most common target of the published metadata is the OAI tables that are accessible for harvesting by external systems. Rosetta supports all of the OAI verbs and request types as they are specified at the following OAI reference site:

Open Archives Initiative Protocol for Metadata Harvesting

In addition, users can configure Rosetta to export the published metadata to a designated folder in the file system, so that other systems can harvest from it. This publishing target can be useful for handling large amounts of data, such as full text versions of IEs.

The published objects are available through HTTP by calling Rosetta in the following way:

```
http://server-name/oaiprovider/
request?verb=ListRecords&metadataPrefix=oai_dc&set=Set&from=2011-08-
22T00:00:01Z
```

The following are the parameters of the OAI HTTP request:

- **Server-name** – holds the server name.

- **Verb** – holds the OAI verb (should be `ListRecords`)
- **MetadataPrefix** – should be `oai_dc`
- **Set** – should be the `Set Spec` from the publishing configuration
- **From** – the `from date` when specifying a date range
- **To** – (optional) the `To` date when specifying a date range

The following is an example of an OAI request:

```
HYPERLINK "http://rosetta.exlibrisgroup.com:1801/oaiprovider/
request?verb=ListRecords&metadataPrefix=oai_dc&set=Set&from=2
011-08-22T00:00:01Z" http:// rosetta.exlibrisgroup.com:1801/
oaiprovider/
request?verb=ListRecords&metadataPrefix=oai_dc&set=Set&from=2
011-08-22T00:00:01Z
```

Data Flow

The following figure illustrates the data flow:

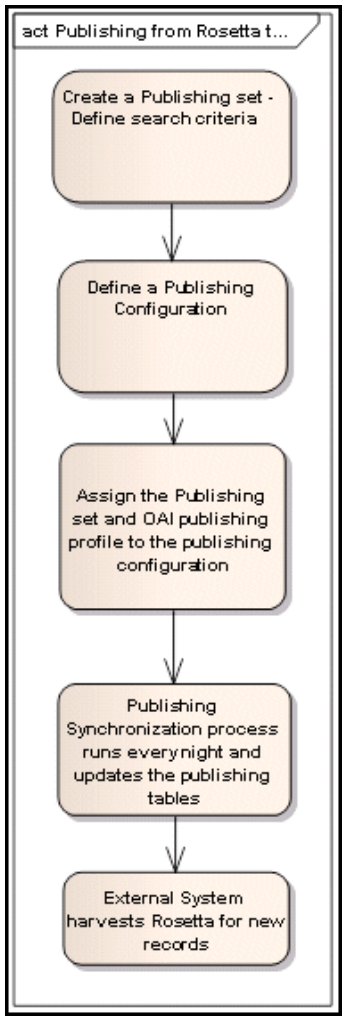

Figure 2: Publishing Data Flow

4

DOI Integration

This section includes:

Introduction

Rosetta supports the storing, generating, and publishing of DOIs (Digital Object Identifiers) in IEs. This allows external systems that store and resolve links to digital objects to store information about IEs in Rosetta.

Current Integrations

Rosetta supports the following DOI integrations:

- **Generic solution** – The generic solution uses a task that is based on a plug-in called PIGenericGenerator that can be configured to receive any kind of string prefix and then generate a unique number. Combined together, this string is stored as a DOI. In order to publish this DOI, the generic solution uses OAI for publishing the DOI along with any other information derives from the IE metadata.
- **Handle** – Integrations with the Handle DOI are used by NLNZ. It contains the following tasks:
  - **Handle creation** – A process-automation task goes over a list of rules and matches the Handle generation profile to the IE based on its DNX fields.

- **Handle publishing** – This is similar to the Handle creation. A process-automation task goes over a list of rules and matches the Handle publishing profile to the IE based on its DNX fields. Publishing Profiles can be a secret key or a private/public key.

- **NLB PID** – To support the integration with the DOI that is used by the National Library Board of Singapore, users can run a process-automation task that interacts with an external server. This task uses a plug-in that generates a unique PID and stores it in the external system, along with the details of the processed IE.

All these integrations are using tasks (each integration has its own task) that can be added to the Enrichment task chain. These tasks can also be performed as a stand-alone task chain on a set of IEs or a single IE (as a service) which are already in the permanent.

Storing the DOI

Once the DOI is generated for each IE, its value is stored in the METS file in a DNX section called **ObjectIdentifier**. This section contains the following fields:

- **objectIdentifierType** – holds the type (such as **urn:nbn**)
- **objectIdentifierValue** – holds the DOI valueof the same type

If the DOI was generated outside of Rosetta, the submission application should populate this DNX section.

Rosetta will not validate a DOI that was generated outside of Rosetta.

NOTE:
If the DOI generation task is performed and a DOI is already in the ObjectIdentifier section, the task will not overwrite it and skip processing the IE.

URN and xepicur Implementation

Rosetta integrates with the DNB (National Library of Germany) for publishing the URN DOI for each IE in Rosetta, along with the Delivery URL of that IE. The XML format that the DNB expects to harvest is called xepicur.

For more details about the URN and xepicur integration see the *URN and X-Epicur Support in Rosetta* document.

Configuration of DOI Integration

For information about configuring the DOI tasks, see the chapter "Digital Object Identifiers" in the *Rosetta Configuration Guide*.