OAI 2.0 Server

Introduction

Open Archives Initiative Protocol for Metadata Harvesting is a low-barrier mechanism for repository interoperability. Data Providers are repositories that expose structured metadata via OAI-PMH. Service Providers then make OAI-PMH service requests to harvest that metadata. OAI-PMH is a set of six verbs or services that are invoked within HTTP.

What is OAI 2.0?

OAI 2.0 is a Java implementation of an OAI-PMH data provider interface (originally developed by Lyncode) that uses XOAI, an OAI-PMH Java Library.

Why OAI 2.0?

Projects like OpenAIRE have specific metadata requirements (to the published content through the OAI-PMH interface). As the OAI-PMH protocol doesn't establish any frame to these specifics, OAI 2.0 can, in a simple way, have more than one instance of an OAI interface (feature provided by the XOAI core library) so one could define an interface for each project. That is the main purpose, although, OAI 2.0 allows much more than that.

Concepts (XOAI Core Library)

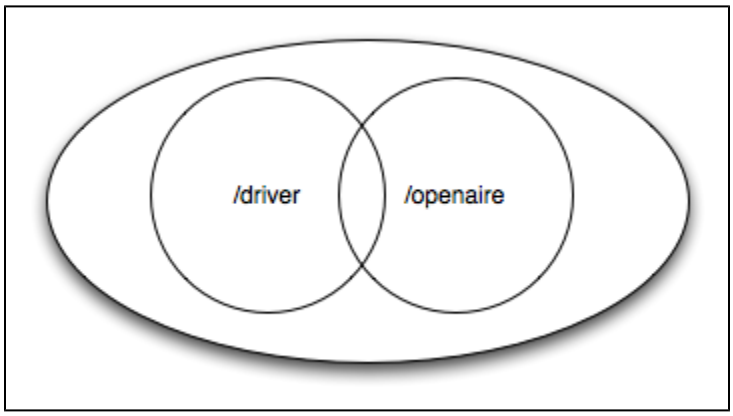
To understand how XOAI works, one must understand the concept of Filter, Transformer and Context. With a Filter it is possible to select information from the data source. A Transformer allows one to make some changes in the metadata before showing it in the OAI interface. XOAI also adds a new concept to the OAI-PMH basic specification, the concept of context. A context is identified in the URL:

```
http://www.example.com/oai/<context>
```

Contexts could be seen as virtual distinct OAI interfaces, so with this one could have things like:

- http://www.example.com/oai/request
- http://www.example.com/oai/driver
- http://www.example.com/oai/openaire

With this ingredients it is possible to build a robust solution that fulfills all requirements of *Driver*, *OpenAIRE* and also other project-specific requirements. As shown in Figure 1, with contexts one could select a subset of all available items in the data source. So when entering the *OpenAIRE* context, all OAI-PMH request will be restricted to that subset of items.



At this stage, contexts could be seen as sets (also defined in the basic OAI-PMH protocol). The magic of XOAI happens when one need specific metadata format to be shown in each context. Metadata requirements by *Driver* slightly differs from the *OpenAIRE* ones. So for each context one must define its specific transformer. So, contexts could be seen as an extension to the concept of sets.

To implement an OAI interface from the XOAI core library, one just need to implement the datasource interface.

OAI 2.0

OAI 2.0 is deployed as a part of the DSpace server (backend) webapp. OAI 2.0 has a configurable data source, by default it will not query the DSpace SQL database at the time of the OAI-PMH request. Instead, it keeps the required metadata in its Solr index (currently in a separate "oai" Solr core) and serves it from there. It's also possible to set OAI 2.0 to only use the database for querying purposes if necessary, but this decreases performance significantly. Furthermore, it caches the requests, so doing the same query repeatedly is very fast. In addition to that it also compiles DSpace items to make uncached responses much faster.

Details about OAI 2.0 internals can be found here.

The OAI 2.0 Server only uses Solr for its indexing. The previous capability to use Database indexing has been removed.

Indexing OAI content

OAI 2.0 uses Solr for all indexing of content.

The Solr index can be updated at your convenience, depending on how fresh you need the information to be. Typically, the administrator sets up a nightly cron job to update the Solr index from the SQL database.

OAI Manager

OAI manager is a utility that allows one to do certain administrative operations with OAI. You can call it from the command line using the dspace launcher:

Syntax

[dspace]/bin/dspace oai <action> [parameters]

Actions

- import *Imports DSpace items into OAI Solr index (also cleans OAI cache)*
- clean-cache *Cleans the OAI cache*

Parameters

- -c *Clears the Solr index before indexing (it will import all items again)*
- -v *Verbose output*
- -h *Shows an help text*

Scheduled Tasks

In order to refresh the OAI Solr index, it is required to run the `[dspace]/bin/dspace oai import` command periodically. You can add the following task to your crontab:

```
0 3 * * * [dspace]/bin/dspace oai import
```

Note that `[dspace]` should be replaced by the correct value, that is, the value defined in `dspace.cfg` parameter `dspace.dir`.

Client-side stylesheet

The OAI-PMH response is an XML file. While OAI-PMH is primarily used by harvesting tools and usually not directly by humans, sometimes it can be useful to look at the OAI-PMH requests directly - usually when setting it up for the first time or to verify any changes you make. For these cases, XOAI provides an XSLT stylesheet to transform the response XML to a nice looking, human-readable and interactive HTML. The stylesheet is linked from the XML response and the transformation takes place in the user's browser (this requires a recent browser, older browsers will only display the XML directly). Most automated tools are interested only in the XML file itself and will not perform the transformation. If you want, you can change which stylesheet will be used by placing it into the `[dspace]/webapps/oai/static` directory (or into the `[dspace-src]/dspace-xoai/dspace-xoai-webapp/src/main/webapp/static` after which you have to rebuild DSpace), modifying the "stylesheet" attribute of the "Configuration" element in `[dspace]/config/crosswalks/oai/xoai.xml` and restarting your servlet container.

Metadata Formats

By default OAI 2.0 provides 12 metadata formats within the /request context:

1. OAI_DC
2. DIDL
3. DIM
4. ETDMS
5. METS
6. MODS
7. OAI-ORE
8. QDC
9. RDF
10. MARC
11. UKETD_DC
12. XOAI

At /driver context it provdes:

1. OAI_DC
2. DIDL
3. METS

And at /openaire context it provides:

1. OAI_DC
2. METS

Encoding problems

There are two main potential sources of encoding problems:

a) The servlet connector port has to use the correct encoding. E.g. for Tomcat, this would be `<Connector port="8080" ... URIEncoding="UTF-8" />`, where the port attribute specifies port of the connector that DSpace is configured to access Solr on (this is usually 8080, 80 or in case of AJP 8009).

b) System locale of the dspace command line script that is used to do the oai import. Make sure the user account launching the script (usually from cron) has the correct locale set (e.g. en_US.UTF-8). Also make sure the locale is actually present on your system.

Configuration

Basic Configuration

Configuration File:	**`[dspace]/config/modules/oai.cfg`**
Property:	oai.enabled
Example Value:	`oai.enabled = true (default)`
Information Note:	Allows you to enable or disable the OAI module/endpoint
Property:	oai.path
Example Value:	`oai.path = oai (default)`
Information Note:	Allows you to specify the path where the OAI module will be deployed. This path is relative to the dspace.server.url. So, for example, if "dspace.server.url=http://localhost:8080/server", then by default the OAI module is available at http://localhost:8080/server/oai/
Property:	`oai.storage`
Example Value:	`oai.storage = solr`
Information Note:	This allows to choose the OAI data source between solr and database. ONLY "solr" is supported at this time.

Property:	`oai.solr.url`
Example Value:	`oai.solr.url = ${solr.server}/oai`
Informational Note:	Solr Server location
Property:	`oai.identifier.prefix`
Example Value:	`oai.identifier.prefix = ${dspace.hostname}`
Informational Note:	OAI persistent identifier prefix. Format - oai:PREFIX:HANDLE
Property:	`oai.config.dir`
Example Value:	`oai.config.dir = ${dspace.dir}/config/crosswalks/oai`
Informational Note:	Configuration directory, used by XOAI (core library). Contains xoai.xml, metadata format XSLTs and transformer XSLTs.
Property:	`oai.cache.enabled`
Example Value:	`oai.cache.enabled = true`
Informational Note:	Whether to enable the OAI cache. Default is true (for better performance).
Property:	`oai.cache.dir`
Example Value:	`oai.cache.dir = ${dspace.dir}/var/oai`
Informational Note:	Directory to store runtime generated files (for caching purposes).

Advanced Configuration

OAI 2.0 allows you to configure following advanced options:

- Contexts
- Transformers
- Metadata Formats
- Filters
- Sets

It's an XML file commonly located at: **[dspace]/config/crosswalks/oai/xoai.xml**

General options

These options influence the OAI interface globally. "per page" means per request, next page (if there is one) can be requested using resumptionToken provided in current page.

- identation [boolean] - whether the output XML should be indented to make it human-readable
- maxListIdentifiersSize [integer] - how many identifiers to show per page (verb=ListIdentifiers)
- maxListRecordsSize [integer] - how many records to show per page (verb=ListRecords)
- maxListSetsSize [integer] - how many sets to show per page (verb=ListSets)
- stylesheet [relative file path] - an xsl stylesheet used by client's web browser to transform the output XML into human-readable HTML

Their location and default values are shown in the following fragment:

```
<Configuration xmlns="http://www.lyncode.com/XOAIConfiguration"
        identation="false"
        maxListIdentifiersSize="100"
        maxListRecordsSize="100"
        maxListSetsSize="100"
        stylesheet="static/style.xsl">
```

Add/Remove Metadata Formats

Each context could have its own metadata formats. So to add/remove metadata formats to/from it, just need add/remove its reference within xoai.xml, for example, imagine one need to remove the XOAI schema from:

```
<Context baseurl="request">
  <Format refid="oaidc" />
  <Format refid="mets" />
  <Format refid="xoai" />
  <Format refid="didl" />
  <Format refid="dim" />
  <Format refid="ore" />
  <Format refid="rdf" />
  <Format refid="etdms" />
  <Format refid="mods" />
  <Format refid="qdc" />
  <Format refid="marc" />
  <Format refid="uketd_dc" />
</Context>
```

Then one would have:

```
<Context baseurl="request">
  <Format refid="oaidc" />
  <Format refid="mets" />
  <Format refid="didl" />
  <Format refid="dim" />
  <Format refid="ore" />
  <Format refid="rdf" />
  <Format refid="etdms" />
  <Format refid="mods" />
  <Format refid="qdc" />
  <Format refid="marc" />
  <Format refid="uketd_dc" />
</Context>
```

It is also possible to create new metadata format by creating a specific XSLT for it. All already defined XSLT for DSpace can be found in the **[dspace]/config/crosswalks/oai/metadataFormats** directory. So after producing a new one, add the following information (location marked using brackets) inside the *<Formats>* element in [dspace]/config/crosswalks/oai/xoai.xml:

```
<Format id="[IDENTIFIER]">
  <Prefix>[PREFIX]</Prefix>
  <XSLT>metadataFormats/[XSLT]</XSLT>
  <Namespace>[NAMESPACE]</Namespace>
  <SchemaLocation>[SCHEMA_LOCATION]</SchemaLocation>
</Format>
```

where:

Parameter	Description
IDENTIFIER	The identifier used within context configurations to reference this specific format, must be unique within all Metadata Formats available.
PREFIX	The prefix used in OAI interface (metadataPrefix=PREFIX).
XSLT	The name of the XSLT file within **[dspace]/config/crosswalks/oai/metadataFormats** directory
NAMESPACE	XML Default Namespace of the created Schema
SCHEMA_LOCATION	URI Location of the XSD of the created Schema

NOTE: Changes in [dspace]/config/crosswalks/oai/xoai.xml requires reloading/restarting the servlet container.

Add/Remove Metadata Fields

The internal DSpace fields (Dublin Core) are exposed in the internal XOAI format (xml). All other metadata formats exposed via OAI are mapped from this XOAI format using XSLT (xoai.xsl itself is just an identity transformation). These XSLT stylesheets are found in the **[dspace]/config/crosswalks/oai/metadataFormats** directory. So e.g. oai_dc.xsl is a transformation from the XOAI format to the oai_dc format (unqualified Dublin Core).

Therefore exposing any DSpace metadata field in any OAI format is just a matter of modifying the corresponding output format stylesheet (This assumes the general knowledge of how XSLT works. For a tutorial, see e.g. http://www.w3schools.com/xsl/).

For example, if you have a DC field "local.note.librarian" that you want to expose in oai_dc as <dc:note> (please note that this is not a valid DC field and thus breaks compatibility), then edit oai_dc.xsl and add the following lines just above the closing tag </oai_dc:dc>:

```
<xsl:for-each select="doc:metadata/doc:element[@name='local']/doc:element[@name='note']/doc:element/doc:element
/doc:field[@name='librarian']">
    <dc:note><xsl:value-of select="." /></dc:note>
</xsl:for-each>
```

If you need to add/remove metadata fields, you're changing the output format. Therefore it is recommended to create a new metadata format as a copy of the one you want to modify. This way the old format will remain available along with the new one and any upgrades to the original format during DSpace upgrades will not overwrite your customizations. If you need the format to have the same name as the original format (e.g. the default oai_dc format), you can create a new context in xoai.xsl containing your modified format with the original name, which will be available as /oai/context-name.

NOTE: Please, keep in mind that the OAI provider caches the transformed output, so you have to run `[dspace]/bin/dspace oai clean-cache` after any .xsl modification and reload the OAI page for the changes to take effect. When adding/removing metadata formats, making changes in [dspace]/config /crosswalks/oai/xoai.xml requires reloading/restarting the servlet container.

Driver/OpenAIRE compliance

The default OAI 2.0 installation provides two new contexts. They are:

- Driver context, which only exposes Driver compliant items;
- OpenAIRE context, which only exposes OpenAIRE compliant items;

However, in order to be exposed DSpace items must be compliant with Driver/OpenAIRE guide-lines.

Driver Compliance

DRIVER Guidelines for Repository Managers and Administrators on how to expose digital scientific resources using OAI-PMH and Dublin Core Metadata, creating interoperability by homogenizing the repository output. The OAI-PMH `driver` set is based on DRIVER Guidelines 2.0.

This set is used to expose items of the repository that are available for open access. It's not necessary for all the items of the repository to be available for open access.

What specific metadata values are expected?

To have items in this set, you must configure your `input-forms.xml` file in order to comply with the DRIVER Guidelines:

- Must have a publication date - dc.date.issued (already configured in DSpace items)
- dc.language must use ISO639-3
- the value of dc.type must be one of the 16 types named in the guidelines

How do you easily add those metadata values?

As DRIVER guidelines use Dublin Core, all the needed items are already registered in DSpace. You just need to configure the deposit process.

OpenAIRE compliance

For OpenAIRE v4 compliance, see OpenAIRE4 Guidelines Compliancy

The OpenAIRE Guidelines 2.0 provide the OpenAIRE compatibility to repositories and aggregators. By implementing these Guidelines, repository managers are facilitating the authors who deposit their publications in the repository in complying with the EC Open Access requirements. For developers of repository platforms, the Guidelines provide guidance to add supportive functionalities for authors of EC-funded research in future versions.

The name of the set in OAI-PMH is "ec_fundedresources" and will expose the items of the repository that comply with these guidelines. These guidelines are based on top of DRIVER guidelines. See version 2.0 of the Guidelines.

See the Application Profile of OpenAIRE.

What specific metadata values are expected?

These are the OpenAIRE metadata values only, to check these and driver metadata values check page 11 of the OpenAIRE guidelines 2.0.

- dc:relation with the project ID (see p.8)
- dc:rights with the access rights information from vocabulary (possible values here)

Optionally:

- dc:date with the embargo end date (recommended for embargoed items)

```
<dc:date>info:eu-repo/date/embargoEnd/2011-05-12<dc:date>
```

How do you easily add those metadata values?

- Have a dc:relation field in `input-forms.xml` with a list of the projects. You can also use the OpenAIRE Authority Control Addon to facilitate the process of finding the project.
- Just use a combo-box for dc:rights to input the 4 options:
  - info:eu-repo/semantics/closedAccess
  - info:eu-repo/semantics/embargoedAccess
  - info:eu-repo/semantics/restrictedAccess
  - info:eu-repo/semantics/openAccess
- Use an input-box for dc:date to insert the embargo end date

Relevant Links

- OAI 2.0 is a standard part of DSpace 3.0
- Download & Install OAI 2.0 for DSpace 1.8.x: http://www.lyncode.com/dspace/addons/xoai/

Sanity check your OAI interface with the OAI Validator

There is a very useful validator for OAI interfaces available at http://validator.oaipmh.com, we urge you to use this validator to confirm your OAI interface is in fact usable.