Configuring OAI-PMH in DSpace

1. OAI Protocol for Metadata Harvesting

Introduction

This section provides a summary of Open Archives Initiative's specifications of the metadata harvesting protocol: "*The Open Archives Initiative Protocol for Metadata Harvesting Protocol v 2.0"*, by Carl Lagoze, Herbert Van de Sompel, Michael Nelson and Simeon Warner[1].

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) provides an application-independent interoperability framework based on metadata harvesting. There are two classes of participants in the OAI-PMH framework:

- Data Providers - systems that support the OAI-PMH as a means of exposing metadata;
- Service Providers - use metadata harvested via the OAI-PMH as a basis for building value-added services.

Terminology

A **harvester** is a client application that issues OAI-PMH requests. A harvester is operated by a service provider as a means of collecting metadata from repositories.

Selective harvesting allows harvesters to limit harvest requests to portions of the metadata available from a repository. The OAI-PMH supports selective harvesting with two types of harvesting criteria that may be combined in an OAI-PMH request: datestamps and set membership. This document will not go into detail on selective harvesting as it is primarily important when implementing a harvester application.

A **resource** is the object or that the metadata describes. The nature of a resource, whether it is physical or digital, or whether it is stored in the repository or is a constituent of another database, is outside the scope of the OAI-PMH.

An **item** is a constituent of a repository from which metadata about a resource can be disseminated. That metadata may be disseminated on-the-fly from the associated resource, cross-walked from some canonical form, actually stored in the repository, etc.

A **unique identifier** unambiguously identifies an item within a repository; the unique identifier is used in OAI-PMH requests for extracting metadata from the item. Items may contain metadata in multiple formats.

A **record** is metadata in a specific metadata format. A record is returned as an XML-encoded byte stream in response to a protocol request. A record has the following components:

- the *header part*: contains the unique identifier of the item and properties necessary for selective harvesting. The header consists of the following parts:
  - the unique identifier: the unique identifier of an item in a repository;
  - the datestamp: the date of creation, modification or deletion of the record;
  - setSpec elements: the set membership of the item;
  - status attribute (optional): indicates the withdrawal of availability of the specified metadata format for the item.
- the *metadata part:* a single manifestation of the metadata from an item. At a minimum, repositories must be able to return records with metadata expressed in the Dublin Core format, without any qualification. Optionally, a repository may also disseminate other formats of metadata.
- the *about part*: an optional and repeatable container to hold data about the metadata part of the record. Two common uses of about containers are:
  - rights statements: to attach terms of use to the metadata they make available through the OAI-PMH.
  - provenance statements: to indicate the provenance of a metadata record, e.g. whether it has been harvested itself and if so from which repository, and when.

[1] http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm

A **deleted record** is a record that is no longer available in the repository's OAI interface. Like a normal record, a deleted record is identified by a unique identifier, a metadataPrefix and a datestamp. Repositories must declare one of three levels of support for deleted records in the deletedRecord element of the Identify response:

- no - the repository does not maintain information about deletions. A repository that indicates this level of support must not reveal a deleted status in any response.
- persistent - the repository maintains information about deletions with no time limit. A repository that indicates this level of support must persistently keep track of the full history of deletions and consistently reveal the status of a deleted record over time.
- transient - the repository does not guarantee that a list of deletions is maintained persistently or consistently. A repository that indicates this level of support may reveal a deleted status for records.

A **set** is an optional construct for grouping items for the purpose of selective harvesting. Repositories may organize items into sets. Set organization may be flat, i.e. a simple list, or hierarchical. Multiple hierarchies with distinct, independent top-level nodes are allowed.

Protocol

OAI-PMH requests are expressed as HTTP requests. A typical implementation uses a standard Web server that is configured to dispatch OAI-PMH requests to the software handling these requests.

HTTP Request Format

OAI-PMH request in a HTTP GET URL

URLs for GET requests have keyword arguments appended to the base URL, separated from it by a question mark [?]. For example, the URL of a GetRecord request to a repository with base URL that is http://an.oa.org/OAI-script might be:

```
http://an.oa.org/OAI-script?verb=GetRecord&identifier=oai:arXiv.org:hep-th/9901001&metadataPrefix=oai_dc
```

However, since special characters in URIs must be encoded, the correct form of the above GET request URL is:

```
http://an.oa.org/OAI-script?verb=GetRecord&identifier=oai%3AarXiv.org%3Ahep-th%2F9901001&metadataPrefix=oai_dc
```

Encoding an OAI-PMH request in an HTTP POST

Keyword arguments are carried in the message body of the HTTP POST, with the format of the POST being:

```
POST http://an.oa.org/OAI-script HTTP/1.0
Content-Length: 82
Content-Type: application/x-www-form-urlencoded
verb=GetRecord&identifier=oai%3AarXiv.org%3Ahep-th%2F9901001&metadataPrefix=oai_dc
```

XML Response Format

Responses to OAI-PMH requests have the following common markup:

The first tag output is an XML declaration where the version is always 1.0 and the encoding is always UTF-8, eg: `<?xml version="1.0" encoding="UTF-8" ?>`

The remaining content is enclosed in a root element `<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">`.

This element must have three attributes that define the XML namespaces used in the remainder of the response and the location of the validating schema:

- `xmlns` - the namespace URI of the OAI-PMH.

- `xmlns:xsi` - the namespace URI for XML schema.
- `xsi:schemaLocation` - the first part of which is the namespace URI of the OAI-PMH, the second part is the URL of the XML schema for validation of the response.

For all responses, the first two children of the root element are:
- `responseDate` - a UTCdatetime indicating the time and date that the response was sent. The UTCdatetime must be uniformly encoded using ISO8601[2].
- `request` - indicating the protocol request that generated this response.
- The third child of the root element is either:
  - an element with the same name as the verb of the respective OAI-PMH request;
  - an `error` element that must be used in case of an error or exception condition.

Examples of OAI-PMH responses can be found in “Requests and Responses” later in this document.

Resumption Tokens

A repository must include a `resumptionToken` element as part of each response that includes an incomplete list. In order to retrieve the next portion of the list, the next request must use the value of that resumptionToken element as the value of the resumptionToken argument of the request. The response containing the incomplete list that completes the list must include an empty resumptionToken element. For an example please refer to the “ListRecords” example below.

Requests and Responses

This section lists the requests, or verbs, defined in the OAI-PMH and a brief summary of their meaning.

GetRecord

This verb is used to retrieve an individual metadata record from a repository.

Arguments

- `identifier` - the unique identifier of the repository item from which the record must be disseminated. (required)
- `metadataPrefix` - metadataPrefix of the format that should be included in the metadata part of the returned record. (required)

Example

Request:

```
http://dspace.mit.edu/oai/request?verb=GetRecord&identifier=oai:dspace.mit.edu:
1721.1/659&metadataPrefix=oai_dc
```

Response:

```
<?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>2009-09-02T12:21:13Z</responseDate>
<request identifier="oai:dspace.mit.edu:1721.1/659" metadataPrefix="oai_dc"
verb="GetRecord">http://dspace.mit.edu/oai/request</request>
<GetRecord>
  <record>
  <header>
  <identifier>oai:dspace.mit.edu:1721.1/659</identifier>
  <datestamp>2006-10-14T12:18:18Z</datestamp>
  <setSpec>hdl_1721.1_1792</setSpec>
  </header>
  <metadata>
    <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
```

[2] http://www.w3.org/TR/NOTE-datetime

```
xmlns:dc="http://purl.org/dc/elements/1.1/" 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:creator>Hatakenaka, Sachi</dc:creator>
      <dc:creator>Rudolph, Jenny</dc:creator>
      <dc:creator>Carroll, John S.</dc:creator>
      <dc:date>2002-05-03T14:28:10Z</dc:date>
      <dc:date>2002-05-03T14:28:10Z</dc:date>
      <dc:date>2002-05-03T14:28:45Z</dc:date>
      <dc:identifier>http://hdl.handle.net/1721.1/659</dc:identifier>
      <dc:description>High-hazard or high-reliability organizations are ideal for the
study of organizational learning processes because of their intense mindfulness regarding
problems. ...</dc:description>
      <dc:language>en_US</dc:language>
      <dc:relation>MIT Sloan School of Management Working Paper;4360-02</dc:relation>
      <dc:subject>organizational change</dc:subject>
      <dc:subject>organizational learning</dc:subject>
      <dc:subject>high-hazard industries</dc:subject>
      <dc:subject>nuclear power plants</dc:subject>
      <dc:subject>management of change</dc:subject>
      <dc:title>Problem Investigation in High-Hazard Industries: Creating and
Negotiational Learning</dc:title>
    </oai_dc:dc>
  </metadata>
</record>
</GetRecord>
</OAI-PMH>
```


Identify

This verb is used to retrieve information about a repository. Some of the information returned is required as part of the OAI-PMH and additional descriptive information. The Identify verb has no parameters.

Example

Request:

```
http://dspace.mit.edu/oai/request?verb=Identify
```

Response:


```
<OAI-PMH>
<responseDate>2009-09-02T12:49:06Z</responseDate>
<request verb="Identify">http://dspace.mit.edu/oai/request</request>
<Identify>
       <repositoryName>DSpace at MIT</repositoryName>
       <baseURL>http://dspace.mit.edu/oai/request</baseURL>
       <protocolVersion>2.0</protocolVersion>
       <adminEmail>dspace-help@mit.edu</adminEmail>
       <earliestDatestamp>2001-01-01T00:00:00Z</earliestDatestamp>
       <deletedRecord>persistent</deletedRecord>
       <granularity>YYYY-MM-DDThh:mm:ssZ</granularity>
       <compression>gzip</compression>
       <compression>deflate</compression>
       <description>
       <toolkit xmlns="http://oai.dlib.vt.edu/OAI/metadata/toolkit"
xsi:schemaLocation="http://oai.dlib.vt.edu/OAI/metadata/toolkit http://oai.dlib.vt.edu/
OAI/metadata/toolkit.xsd">
              <title>OCLC's OAICat Repository Framework</title>
              <author>
                     <name>Jeffrey A. Young</name>
                     <email>jyoung@oclc.org</email>
                     <institution>OCLC</institution>
              </author>
              <version>1.5.48</version>
              <toolkitIcon>http://alcme.oclc.org/oaicat/oaicat_icon.gif</toolkitIcon>
              <URL>http://www.oclc.org/research/software/oai/cat.shtm</URL>
       </toolkit>
       </description>
</Identify>
</OAI-PMH>
```

ListIdentifiers

This verb retrieves only headers rather than records. Depending on the repository's support for deletions, a returned header may have a status attribute of "deleted".

Arguments

- `from` - a UTCdatetime value, which specifies a lower bound for datestamp-based selective harvesting. (optional)
- `until` - a UTCdatetime value, which specifies a upper bound for datestamp-based selective harvesting. (optional)
- `metadataPrefix` - specifies that headers should be returned only if the metadata format matching the supplied metadataPrefix is available or, depending on the repository's support for deletions, has been deleted. The metadata formats supported by a repository and for a particular item can be retrieved using the ListMetadataFormats request.
- `set` an optional argument with a setSpec value , which specifies set criteria for selective harvesting.
- `resumptionToken` an exclusive argument with a value that is the flow control token returned by a previous ListIdentifiers request that issued an incomplete list.

Example

Request:

```
http://dspace.mit.edu/oai/request?
verb=ListIdentifiers&metadataPrefix=oai_dc&set=hdl_1721.1_7652&from=2009-01-01
```

Response:

```
<OAI-PMH>
<responseDate>2009-09-02T12:42:42Z</responseDate>
<request metadataPrefix="oai_dc" verb="ListIdentifiers" from="2009-01-01"
set="hdl_1721.1_7652">http://dspace.mit.edu/oai/request</request>
<ListIdentifiers>
       <header>
              <identifier>oai:dspace.mit.edu:1721.1/44197</identifier>
              <datestamp>2009-01-27T07:02:51Z</datestamp>
              <setSpec>hdl_1721.1_7652</setSpec>
              <setSpec>hdl_1721.1_7802</setSpec>
       </header>
       <header>
              <identifier>oai:dspace.mit.edu:1721.1/44199</identifier>
              <datestamp>2009-01-27T07:03:18Z</datestamp>
              <setSpec>hdl_1721.1_7652</setSpec>
              <setSpec>hdl_1721.1_7802</setSpec>
       </header>
</ListIdentifiers>
</OAI-PMH>
```

ListMetadataFormats

This verb is used to retrieve the metadata formats available from a repository. An optional argument restricts the request to the formats available for a specific item.

Arguments

- `identifier` - specifies the unique identifier of the item for which available metadata formats are being requested. If this argument is omitted, then the response includes all metadata formats supported by this repository. Note that the fact that a metadata format is supported by a repository does not mean that it can be disseminated from all items in the repository. (optional)

Example

Request:

```
http://dspace.mit.edu/oai/request?verb=ListMetadataFormats
```

Response:

```
<OAI-PMH>
<responseDate>2009-09-02T14:14:11Z</responseDate>
<request verb="ListMetadataFormats">http://dspace.mit.edu/oai/request</request>
<ListMetadataFormats>
   <metadataFormat>
       <metadataPrefix>mets</metadataPrefix>
       <schema>http://www.loc.gov/standards/mets/mets.xsd</schema>
       <metadataNamespace>http://www.loc.gov/METS/</metadataNamespace>
   </metadataFormat>
   <metadataFormat>
       <metadataPrefix>rdf</metadataPrefix>
       <schema>http://www.openarchives.org/OAI/2.0/rdf.xsd</schema>
       <metadataNamespace>http://www.openarchives.org/OAI/2.0/rdf/</metadataNamespace>
   </metadataFormat>
   <metadataFormat>
       <metadataPrefix>oai_dc</metadataPrefix>
       <schema>http://www.openarchives.org/OAI/2.0/oai_dc.xsd</schema>
       <metadataNamespace>http://www.openarchives.org/OAI/2.0/oai_dc/</metadataNamespace>
   </metadataFormat>
</ListMetadataFormats>
</OAI-PMH>
```

ListRecords

This verb is used to harvest records from a repository. Optional arguments permit selective harvesting of records based on set membership and/or datestamp. Depending on the repository's support for deletions, a returned header may have a status attribute of "deleted" if a record matching the arguments specified in the request has been deleted.

Arguments

- `from` - UTCdatetime value, specifies a lower bound for datestamp-based selective harvesting. (optional)
- `until` - UTCdatetime value, specifies a upper bound for datestamp-based selective harvesting. (optional)
- `set` - a setSpec value , which specifies set criteria for selective harvesting. (optional)
- `resumptionToken` - a value that is the flow control token returned by a previous ListRecords request that issued an incomplete list. (exclusive)
- `metadataPrefix` - specifies the metadataPrefix of the format that should be included in the metadata part of the returned records. Records should be included only for items from which the metadata format matching the metadataPrefix can be disseminated. The metadata formats supported by a repository and for a particular item can be retrieved using the ListMetadataFormats request. (required, unless the exclusive argument resumptionToken is used)

Example

Request:

```
http://dspace.mit.edu/oai/request?
verb=ListRecords&metadataPrefix=oai_dc&set=hdl_1721.1_7652&from=2009-01-01
```

Response:

```
<OAI-PMH>
<responseDate>2009-09-02T15:32:22Z</responseDate>
<request metadataPrefix="oai_dc" verb="ListRecords" from="2009-08-01"
set="hdl_1721.1_7652">http://dspace.mit.edu/oai/request</request>
<ListRecords>
       <record>
              <header>
                     <identifier>oai:dspace.mit.edu:1721.1/46408</identifier>
                     <datestamp>2009-08-27T06:05:38Z</datestamp>
                     <setSpec>hdl_1721.1_7652</setSpec>
                     <setSpec>hdl_1721.1_7802</setSpec>
              </header>
              <metadata>
                     ...
```

```
            </metadata>
        </record>
        <record>
            <header>
                <identifier>oai:dspace.mit.edu:1721.1/46689</identifier>
                <datestamp>2009-08-27T07:22:55Z</datestamp>
                <setSpec>hdl_1721.1_7652</setSpec>
                <setSpec>hdl_1721.1_7710</setSpec>
                <setSpec>hdl_1721.1_7742</setSpec>
                <setSpec>hdl_1721.1_7802</setSpec>
                <setSpec>hdl_1721.1_7888</setSpec>
                <setSpec>hdl_1721.1_7929</setSpec>
            </header>
            <metadata>
                </metadata>
        </record>
        ...
        ...
        <resumptionToken expirationDate="2009-09-02T16:33:16Z">
            2009-01-01T00:00:00Z/9999-12-31T23:59:59Z/hdl_1721.1_7652/oai_dc/100
        </resumptionToken>
</ListRecords>
</OAI-PMH>
```

Request next Records using resumption token:
`http://dspace.mit.edu/oai/request?verb=ListRecords&resumptionToken=2009-01-01T00:00:00Z/9999-12-31T23:59:59Z/hdl_1721.1_7652/oai_dc/100`

ListSets

This verb is used to retrieve the set structure of a repository, useful for selective harvesting.

Arguments

- `resumptionToken` - a value that is the flow control token returned by a previous ListSets request that issued an incomplete list. (exclusive)

Example

Request:

```
http://dspace.mit.edu/oai/request?verb=ListSets
```

Response:

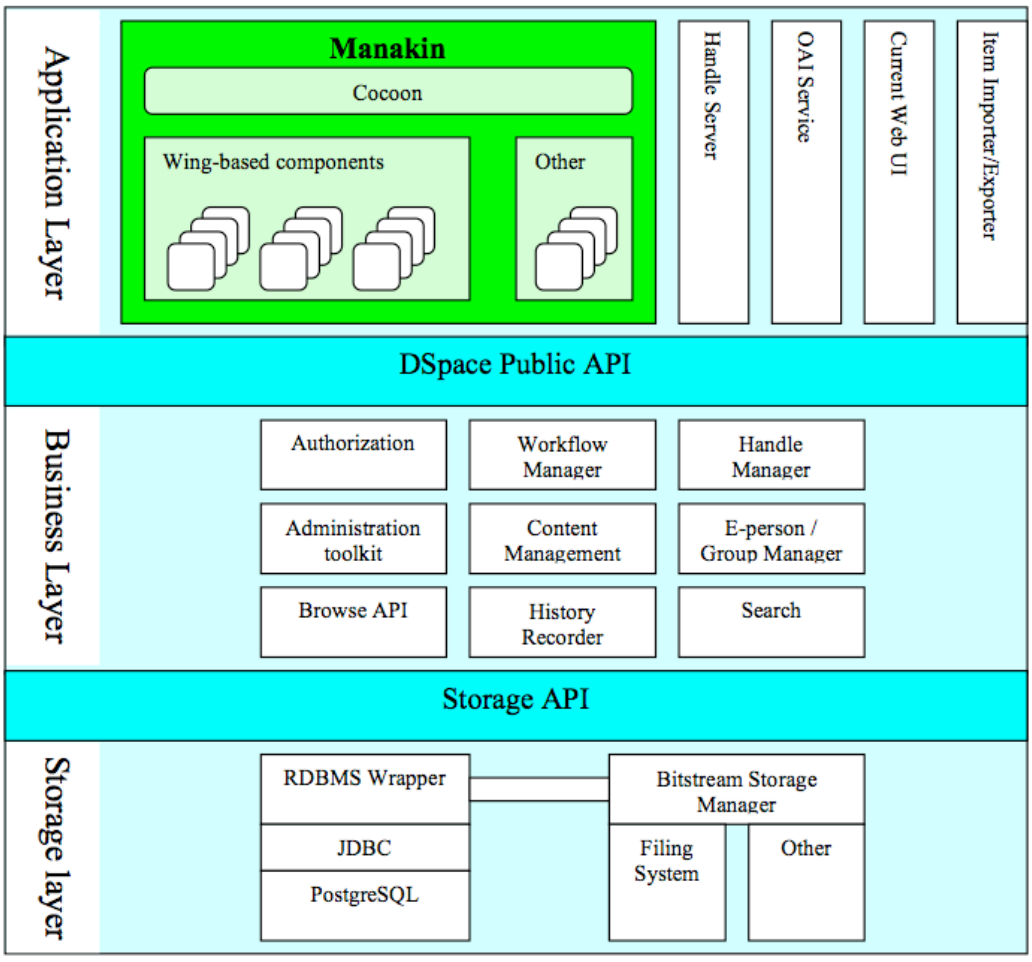
```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://
www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
<responseDate>2009-09-02T12:30:03Z</responseDate>
<request verb="ListSets">http://dspace.mit.edu/oai/request</request>
<ListSets>
        <set>
            <setSpec>hdl_1721.1_33982</setSpec>
            <setName>Civil and Environmental Engineering (1) - Archived</setName>
        </set>
        <set>
            <setSpec>hdl_1721.1_7800</setSpec>
            <setName>Civil and Environmental Engineering - Bachelor's degree</setName>
        </set>
        <set>
            <setSpec>hdl_1721.1_7648</setSpec>
            <setName>Civil and Environmental Engineering - Bachelor's degree</setName>
        </set>
        <set>
            <setSpec>hdl_1721.1_7801</setSpec>
            <setName>Civil and Environmental Engineering - Engineer's degree</setName>
        </set>
        <set>
            <setSpec>hdl_1721.1_7649</setSpec>
            <setName>Civil and Environmental Engineering - Engineer's degree</setName>
```

```
        </set>
        <set>
                <setSpec>hdl_1721.1_7802</setSpec>
                <setName>Civil and Environmental Engineering - Master's degree</setName>
        </set>
        <set>
                <setSpec>hdl_1721.1_7652</setSpec>
                <setName>Civil and Environmental Engineering - Master's degree</setName>
        </set>
</ListSets>
</OAI-PMH>
```

2. OAI-PMH in DSpace

OAI-PMH in the DSpace Architecture

The DSpace build process builds a Web application archive, `[dspace]/webapps/oai.war`). This 'webapp' is deployed to receive and respond to OAI-PMH requests via HTTP. In a typical configuration, this is deployed at oai, and the 'base URL' of this DSpace deployment would be: `http://dspace.myu.edu/oai/request`



source: "Manakin Developer's Guide" Scott Phillips et al.[3]

DSpace exposes the Dublin Core metadata for items that are publicly accessible. Additionally, the collection structure is also exposed via the ListSets request. OCLC's open source OAICat[4] framework is used to

[3] http://di.tamu.edu/projects/xmlui/resources/DevelopersGuide.pdf

[4] http://www.oclc.org/research/software/oai/cat.htm

provide this functionality. DSpace's OAI service supports the exposing of deletion information for withdrawn items, but not for items that are expunged. DSpace also supports OAI-PMH resumption tokens. The OAI service can also be configured to use of any crosswalk plugin to offer additional metadata formats, such as MODS. DSpace provides implementations of the OAICat interfaces `AbstractCatalog`, `RecordFactory` and `Crosswalk` that interface with the DSpace content management API and harvesting API (in the search subsystem).

The basic `oai_dc` unqualified Dublin Core, `mets` and `rdf` metadata set exports are enabled by default. When `oai_dc` metadata is harvested, the qualifiers are simply omitted; for example, `description.abstract` is exposed as unqualified `description`. The description.provenance field is hidden, as this contains private information about the submitter and workflow reviewers of the item, including their e-mail addresses. Additionally, in accordance with OAI community practices, values of `contributor.author` are exposed as `creator` values.

Other metadata formats are supported as well, using other Crosswalk implementations; for available crosswalk implementations consult the `oaicat.properties` file described below. To enable a format, simply uncomment the lines beginning with `Crosswalks.*`. Multiple formats are allowed, and the current list includes, in addition to unqualified DC, METS and RDF: MPEG DIDL, MODS, QDC.

Basic Configuration

There are two configuration files relevant to OAI support in DSpace:

- `oaicat.properties`: resides as an editable file in `[dspace-source]/dspace/config`, and the live version is written to `[dspace]/config` (which is not supposed to be edited).
- `oai-web.xml`: standard Java Servlet 'deployment descriptor' is stored in the source as `[dspace-source]/dspace-oai/dspace-oai-webapp/src/main/webapp/WEB-INF/web.xml`, and is written to `dspace-oai/dspace-oai-webapp/src/main/webapp/WEB-INF/web.xml`.

OAI-PMH allows repositories to expose the sets in which records are placed (see the ListRecords example above). A record can be in zero or more sets. DSpace exposes communities and collections as sets. Each community and collection has a corresponding OAI set, discoverable by harvesters via the `ListSets` verb. The `setSpec` is the Handle of the community or collection, with the ':' and '/' converted to underscores so that the Handle is a legal `setSpec`, for example: `hdl_1721.1_1234`. The community or collection name is also the name of the corresponding set.

Unique Identifier

The OAI identifiers that DSpace uses are of the form: `oai:host name:handle`

For example: `oai:dspace.myu.edu:123456789/345`

If you wish to use a different scheme, this can easily be changed by editing the value of OAI_ID_PREFIX at the top of the `org.dspace.app.oai.DSpaceOAICatalog` class. You do not need to change the code if the above scheme works for you; the code picks up the host name and Handles automatically from the DSpace configuration.

Access control

OAI provides no authentication/authorisation details, although these could be implemented using standard HTTP methods.

Depending on the configuration of your repository, either all metadata is publicly available for harvesting, or only items with anonymous READ access are offered in the harvest. In the dspace.cfg file, `harvest.includerestricted.oai` can be set to `true` to expose all metadata of every item, or set to `false` to only expose metadata of the items with anonymous READ access assigned to the item (independent of bitstream permissions). The latter will ensure item display pages which cannot be viewed in the browser are not harvestable either.

One should be weary of protected items that are made public after a time. When this happens, the items are "new" from the OAI-PMH perspective.

Modification Date (OAI Date Stamp)

OAI-PMH harvesters need to know when a record has been created, changed or deleted. DSpace keeps track of a 'last modified' date for each item in the system, and this date is used for the OAI-PMH date stamp. This means that any changes to the metadata (e.g. admins correcting a field, or a withdrawal) will be exposed to harvesters.

Deletions

DSpace keeps track of deletions (withdrawals). These are exposed via OAI-PMH, which has a specific mechanism for dealing with this. Since DSpace keeps a permanent record of withdrawn items, in the OAI-PMH sense, DSpace supports deletions 'persistently'. This is as opposed to 'transient' deletion support, which would mean that deleted records are forgotten after a time.

Once an item has been withdrawn, only the OAI-PMH harvest requests that specify a date range in which the withdrawal occurred will find the 'deleted' record header. Harvest requests that specify a date range prior to the withdrawal will not find the record, despite the fact that the record did exist at that time.

As an example, consider an item that was created on 2002-05-02 and withdrawn on 2002-10-06. A request to harvest the month 2002-10 will yield the 'record deleted' header. However, a harvest of the month 2002-05 will not yield the original record.

Note that currently, the deletion of 'expunged' items is not exposed through OAI-PMH.

Flow Control (Resumption Tokens)

An OAI data provider can prevent the possible high impact on the repository's performance impact caused by harvesting. It does so by forcing a harvester to receive data in time-separated chunks. If the data provider receives a request for a lot of data, it can send part of the data with a resumption token. The harvester can then return later with the resumption token and continue.

DSpace supports resumption tokens for `ListRecords` OAI-PMH requests, as shown in the LIstRecords example above. `ListIdentifiers` and `ListSets` requests do not produce a particularly high load on the system, so in DSpace resumption tokens are not used for those requests.

By default, each `ListRecords` request will return at most 100 records. This limit is set at the top of `org.dspace.app.oai.DSpaceOAICatalog.java` (`MAX_RECORDS`).

When a resumption token is issued, the optional completeListSize and cursor attributes are not included. OAICat sets the expirationDate of the resumption token to one hour after it was issued, though in fact since DSpace resumption tokens contain all the information required to continue a request in a correct manner after the resumption token has been expired, the resumption token in DSpace does not actually expire.

Resumption tokens contain all the state information required to continue a request. The format is:

`from/until/setSpec/offset`

`from` and `until` are the ISO 8601 dates passed in as part of the original request, and `setSpec` is also taken from the original request.

`offset` is the number of records that have already been sent to the harvester.

Example resumption token: `2003-01-01//hdl_1721_1_1234/300`

This means the harvest is 'from' `2003-01-01`, has no 'until' date, is for collection `hdl:1721.1/1234`, and 300 records have already been sent to the harvester. (Actually, if the original OAI-PMH request doesn't specify a 'from' or 'until, OAICat fills them out automatically to '`0000-00-00T00:00:00Z`' and '`9999-12-31T23:59:59Z`' respectively. This means DSpace resumption tokens will always have from and until dates in them.)

Crosswalk Implementations

Introduction

Crosswalks are software modules that translate DSpace object metadata to a specific external representation. Crosswalks can be implemented and configured using different strategies in DSpace: Java Dissemination Crosswalks, XSLT-based Crosswalks and Java OAICat crosswalks.

Java Dissemination Crosswalks

A Java Dissemination Crosswalk interprets DSpace's internal data structure and crosswalks it to the external format. For example, a MODS dissemination crosswalk generates a MODS document from the metadata on a DSpace Item. Dissemination Crosswalk plugins are named plugins, so it is easy to add new crosswalks.

Configuring Dissemination Crosswalks

Dissemination crosswalk plugins are configured as plugins for the interface `org.dspace.content.crosswalk.DisseminationCrosswalk`.

You can add names for existing crosswalks, add new plugin classes, and add new configurations for the configurable crosswalks as noted below.

Configuring MODS Dissemination Crosswalk

The MODS crosswalk is a self-named plugin. To configure this crosswalk, an entry should be added to oaicat.properties stating:

```
Crosswalks.mods=org.dspace.app.oai.PluginCrosswalk
```

To configure an instance of the MODS crosswalk, add a property to the DSpace configuration starting with `crosswalk.mods.properties.`; the final word of the property name becomes the plugin's name. For example, a property name `crosswalk.mods.properties.MODS` defines a crosswalk plugin named "MODS".

The value of this property is a path to a separate properties file containing the configuration for this crosswalk. The pathname is relative to the DSpace configuration directory, i.e. the config subdirectory of the DSpace install directory. So, a line like:

```
crosswalk.mods.properties.MODS = crosswalks/mods.properties
```

defines a crosswalk named MODS whose configuration comes from the file `[dspace]/config/crosswalks/mods.properties`.

The MODS crosswalk properties file is a list of properties describing how DSpace metadata elements are to be turned into elements of the MODS XML output document. The property name is a concatenation of the metadata schema, element name, and optionally the qualifier.

For example, the contributor.author element in the native Dublin Core schema would be: `dc.contributor.author`. The value of the property is a line containing two segments separated by the vertical bar ("|"): The first part is an XML fragment which is copied into the output document. The second is an XPath expression describing where in that fragment to put the value of the metadata element. For example, in this property:

```
dc.contributor.author = <mods:name><mods:role><mods:roleTerm type="text">author</mods:roleTerm></mods:role><mods:namePart>%s</mods:namePart></mods:name> | mods:namePart/text()
```

Read the example configuration file for more details. Some of the examples include the string "%s" in the prototype XML where the text value is to be inserted for human readablility, but don't pay any attention to it, it is an artifact that the crosswalk ignores. For example, given an author named Jack Florey, the crosswalk will insert

```
<mods:name>
       <mods:role>
              <mods:roleTerm type="text">author</mods:roleTerm>
       </mods:role>
       <mods:namePart>Jack Florey</mods:namePart>
</mods:name>
```

into the output document.

Configuring Qualified Dublin Core (QDC) dissemination crosswalk

The QDC crosswalk is a self-named plugin. To configure this crosswalk, an entry should be added to oaicat.properties stating: `Crosswalks.qdc=org.dspace.app.oai.PluginCrosswalk`

To configure an instance of the QDC crosswalk, add a property to the DSpace configuration starting with `crosswalk.qdc.properties.`; the final word of the property name becomes the plugin's name. For example, a property name `crosswalk.qdc.properties.QDC` defines a crosswalk plugin named "QDC".

The value of this property is a path to a separate properties file containing the configuration for this crosswalk. The pathname is relative to the DSpace configuration directory, i.e. the config subdirectory of the DSpace install directory.

So, a line like: `crosswalk.qdc.properties.QDC = crosswalks/qdc.properties` defines a crosswalk named QDC whose configuration comes from the file `[dspace]/config/crosswalks/qdc.properties`.

You will also need to configure the namespaces and schema location strings for the XML output generated by this crosswalk. The namespaces property names are of the format:

```
crosswalk.qdc.namespace.prefix = uri
```

where prefix is the namespace prefix and uri is the namespace URI.

The following shows how a crosswalk named "QDC" would be configured:

```
crosswalk.qdc.properties.QDC = crosswalks/QDC.properties
crosswalk.qdc.namespace.QDC.dc = http://purl.org/dc/elements/1.1/
crosswalk.qdc.namespace.QDC.dcterms = http://purl.org/dc/terms/
crosswalk.qdc.schemaLocation.QDC = http://purl.org/dc/terms/
       http://dublincore.org/schemas/xmls/qdc/2003/04/02/qualifieddc.xsd
```

The QDC crosswalk properties file is a list of properties describing how DSpace metadata elements are to be turned into elements of the Qualified DC XML output document. The property name is a concatenation of the metadata schema, element name, and optionally the qualifier. For example, the contributor.author element in the native Dublin Core schema would be: dc.contributor.author. The value of the property is an XML fragment, the element whose value will be set to the value of the metadata field in the property key.

For example, in this property:

```
dc.coverage.temporal = <dcterms:temporal />
```

the generated XML in the output document would be:

```
<dcterms:temporal>Fall, 2005</dcterms:temporal>
```

XSLT-based crosswalks

The XSLT crosswalks use XSL stylesheet transformation (XSLT) to transform DSpace's internal metadata to an XML-based external metadata format. XSLT crosswalks are much more powerful and flexible than the MODS and QDC crosswalks above, but they require knowledge of XSL. Given that, you can create all the crosswalks you need just by adding stylesheets and configuration lines, without touching any of the Java code.

A XSLT dissemination crosswalk is described by a configuration key starting with 'crosswalk.dissemination.", like `crosswalk.dissemination.PluginName.stylesheet = path`.The

PluginName is, of course, the plugin's name. The path value is the path to the file containing the crosswalk stylesheet (relative to `[dspace]/config`).

You can make two different plugin names point to the same crosswalk, by adding two configuration entries with the same path, e.g.

```
crosswalk.submission.MyFormat.stylesheet = crosswalks/myformat.xslt
crosswalk.submission.almost_DC.stylesheet = crosswalks/myformat.xslt
```

The crosswalk must also be configured with an XML Namespace (including prefix and URI) and an XML Schema for its output format. This is configured on additional properties in the DSpace Configuration, i.e.:

```
crosswalk.dissemination.PluginName.namespace. Prefix = namespace-URI
crosswalk.dissemination.PluginName.schemaLocation = schemaLocation value
```

For example:

```
crosswalk.dissemination.qdc.namespace.dc = http://purl.org/dc/elements/1.1/
crosswalk.dissemination.qdc.namespace.dcterms = http://purl.org/dc/terms/
crosswalk.dissemination.qdc.schemaLocation = \
     http://purl.org/dc/elements/1.1/
     http://dublincore.org/schemas/xmls/qdc/2003/04/02/qualifieddc.xsd
```

DSpace Intermediate Metadata (DIM) format

XSLT crosswalk plugins translate between the external metadata format and an XML format called DSpace Intermediate Metadata, which exists only for the purpose of XSLT crosswalks. It is never to be exported from DSpace, since it is not an acknowledged metadata format, it is simply an expression of the way DSpace stores its metadata fields internally. All the elements in a DIM document are in the namespace http://www.dspace.org/xmlns/dspace/dim.

The root element is named dim. It has zero or more children, all field elements. It may have an attribute dspaceType, which identifies the type of object ("ITEM", "COLLECTION", or "COMMUNITY") this metadata describes. This attribute is only guaranteed to be set for dissemination crosswalks.

Each field element may have the following attributes:
- `mdschema` - The metadata schema, e.g. "dc". (Required)
- `element` - Element name, such as "contributor". (Required)
- `qualifier` - Qualifier name, such as "author".
- `lang` - Language code describing language of this entry.

The value of field is the value of that metadata field. Fields with the same qualifiers may be repeated. Here is an example of the DIM format:

```
<dim:dim xmlns:dim="http://www.dspace.org/xmlns/dspace/dim" dspaceType="ITEM">
  <dim:field mdschema="dc" element="title" lang="en_US">
    The Endochronic Properties of Resublimated Thiotimonline
  </dim:field>
  <dim:field mdschema="dc" element="contributor" qualifier="author">
    Isaac Asimov
  </dim:field>
  <dim:field mdschema="dc" element="language" qualifier="iso">
    eng
  </dim:field>
  <dim:field mdschema="dc" element="subject" qualifier="other" lang="en_US">
    time-travel scifi hoax
  </dim:field>
  <dim:field element="publisher">
    Boston University Department of Biochemistry
  </dim:field>
</dim:dim>
```

Testing XSLT Crosswalks

The XSLT crosswalks will automatically reload an XSL stylesheet that has been modified, so you can edit and test stylesheets without restarting DSpace. You can test a dissemination crosswalk by hooking it up to an OAI-PMH crosswalk and using an OAI request to get the metadata for a known item.

Example DIMtoQDC XSLT crosswalk:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  xmlns:dspace="http://
www.dspace.org/xmlns/dspace/dim"  xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://purl.org/dc/terms/" version="1.0">

<xsl:template match="dspace:dim">
       <xsl:element name="dcterms:qualifieddc">
              <xsl:apply-templates/>
       </xsl:element>
</xsl:template>
<xsl:template match="dspace:field[@element ='title']">
       <xsl:element name="dc:title">
              <xsl:value-of select="text()"/>
       </xsl:element>
</xsl:template>
<xsl:template match="dspace:field[@element ='contributor' and @qualifier='author']">
       <xsl:element name="dc:author">
              <xsl:value-of select="text()"/>
       </xsl:element>
</xsl:template>
<xsl:template match="dspace:field[@element ='contributor' and @qualifier='illustrator']">
       <xsl:element name="dc:author">
              <xsl:value-of select="text()"/>
       </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

Java OAICat Crosswalks

Java OAICat crosswalks are direct implementations of the OAICat Crosswalk class, independent of the DSpace dissemination interface. Examples of these classes are the METS, unqualified Dublin Core, and RDF crosswalk. These crosswalks are Java implementation based (like the Java Dissemination Crosswalks), and only require configuration to be present in the `oaicat.properties` file.

Each of these crosswalks requires a line similar to

```
Crosswalks.rdf=org.dspace.app.oai.RDFCrosswalk
```

to be present in the `oaicat.properties file`.

The two major differences between Java OAICat crosswalks and Java Dissemination crosswalks are:

- the interface whose methods are to be implemented
- the dissemination crosswalks can be used for exporting items using the DSpace packager, while the OAICat crosswalks do not offer this feature.

Example Java OAICat crosswalk implementation

```
public class RDFCrosswalk extends Crosswalk
{
// Base URL for thumbnails
private String baseUrl = null;

// Hostname for rdf URI
private String hostName = null;

// Constructor that sets the baseUrl and hostName
public RDFCrosswalk(Properties properties)

// Iterate over an item' s metadata and outputs the String representation of the records
// In this case the rdf string representation
public String createMetadata(Object nativeItem) throws CannotDisseminateFormatException

// Determines whether or not an item is suitable for crosswalking (in this case to rdf)
    public boolean isAvailableFor(Object nativeItem)
}
```

TERMS OF USE

PLEASE READ THESE TERMS OF USE CAREFULLY BEFORE USING THESE COURSE MATERIALS. By using these course materials, you signify your assent to these terms of use. If you do not agree to these terms of use, please do not use these course materials.

RESTRICTIONS ON USE OF MATERIALS. These course materials are owned by @mire NV, Technologielaan 9, 3001 Heverlee (Belgium).

No components from these course materials owned, licensed or controlled by @mire NV may be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, except that you may download one copy of the materials on any single computer for your personal, non-commercial home use only, provided you keep intact all copyright and other proprietary notices.

Modification of the materials or use of the materials for any other purpose is a violation of @mire's copyright and other proprietary rights. The use of any such material on any other web site or networked computer environment is prohibited.

To request permission to reproduce materials,
call +32 2 888 29 56,
email info@atmire.com,
or write to @mire NV, Technologielaan 9, 3001 Heverlee, Belgium.