OAI-PMH Data Provider 2.0 (Internals)

OAI-PMH Data Provider 2.0 (Internals)

The DSpace platform supports the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) version 2.0 as a data provider. This is accomplished using the XOAI OAI-PMH Java Toolkit.

The DSpace build process builds a Web application archive, *[dspace-source]/build/oai.war*), in much the same way as the Web UI build process described above. The only differences are that the JSPs are not included. This "webapp" is deployed to receive and respond to OAI-PMH requests via HTTP. In a typical configuration, this is deployed at *oai*, containing request, driver and openaire contexts, for example:

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

The "base URL" of this DSpace deployment would be:

```
http://dspace.myu.edu/oai/request
```

But one could also provide the Driver or OpenAIRE contexts:

```
http://dspace.myu.edu/oai/driver
http://dspace.myu.edu/oai/openaire
```

It is this URL that should be registered with www.openarchives.org.

DSpace provides implementations of the XOAI data sources interfaces.

Sets

OAI-PMH allows repositories to expose an hierarchy of sets in which records may be placed. A record can be in zero or more sets.

DSpace exposes collections and communities as sets.

Each community and collection has a corresponding OAI set, discoverable by harvesters via the ListSets verb. The setSpec is based on the community/collection handle, with the "/" converted to underscore to form a legal setSpec. The setSpec is prefixed by "com_" or "col_" for communities and collections, respectively (this is a change in set names in DSpace 3.0 / OAI 2.0). For example:

```
col_1721.1_1234
```

Naturally enough, the community/collection name is also the name of the corresponding set.

Unique Identifier

Every item in OAI-PMH data repository must have an unique identifier, which must conform to the URI syntax. As of DSpace 1.2, Handles are not used; this is because in OAI-PMH, the OAI identifier identifies the *metadata record* associated with the *resource*. The *resource* is the DSpace item, whose *resource identifier* is the Handle. In practical terms, using the Handle for the OAI identifier may cause problems in the future if DSpace instances share items with the same Handles; the OAI metadata record identifiers should be different as the different DSpace instances would need to be harvested separately and may have different metadata for the item.

The OAI identifiers that DSpace uses are of the form:

```
oai:PREFIX: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 identifier.prefix at [dspace]/config/modules/oai.cfg file.

Access control

OAI provides no authentication/authorisation details, although these could be implemented using standard HTTP methods. It is assumed that all access will be anonymous for the time being.

A question is, "is all metadata public?" Presently the answer to this is yes; all metadata is exposed via OAI-PMH, even if the item has restricted access policies. The reasoning behind this is that people who do actually have permission to read a restricted item should still be able to use OAI-based services to discover the content. But, exposed data could be changed by changing the XSLT defined at [dspace]/config/crosswalks/oai/metadataFormats.

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.

"About" Information

As part of each record given out to a harvester, there is an optional, repeatable "about" section which can be filled out in any (XML-schema conformant) way. Common uses are for provenance and rights information, and there are schemas in use by OAI communities for this. Presently DSpace does not provide any of this information, but XOAI core library allows its definition. This requires to dive into code and perform some changes.

Deletions

DSpace keeps track of deletions (withdrawals). These are exposed via OAI, which has a specific mechansim 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, OAI-PMH harvests of the date range in which the withdrawal occurred will find the "deleted" record header. Harvests of 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 of this, 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 presently, the deletion of "expunged" items is not exposed through OAI.

Flow Control (Resumption Tokens)

An OAI data provider can prevent any performance impact caused by harvesting 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", "ListIdentifiers" and "ListSets" OAI-PMH requests.

Each OAI-PMH ListRecords request will return at most 100 records (by default) but it could be configured in the `[dspace]/config/crosswalks/oai/xoai.xml` file.

When a resumption token is issued, the optional *completeListSize* and *cursor* attributes are included. OAI 2.0 resumption tokens are persistent, so *expirationDate* of the resumption token is undefined, they do not expire.

Resumption tokens contain all the state information required to continue a request and it is encoded in Base64.