

OAI

OAI Interfaces

- 1 [OAI-PMH Activation](#)
 - 1.1 [Enabling OAI-PMH Interface](#)
 - 1.2 [OAI-PMH Configuration](#)
 - 1.2.1 [Activating Additional OAI-PMH Crosswalks](#)
 - 1.2.1.1 [DIDL](#)
- 2 [OAI-PMH / OAI-ORE Harvester](#)
 - 2.1 [Harvesting from another DSpace](#)
 - 2.2 [OAI-PMH / OAI-ORE Harvester Configuration](#)
 - 2.3 [Activating / Using the OAI-PMH / OAI-ORE Harvester](#)

OAI-PMH Activation

In the following sections, you will learn how to configure OAI-PMH and activate additional OAI-PMH crosswalks. The user is also referred to [OAI-PMH Data Provider](#) for greater depth details of the program.

The OAI-PMH Interface may be used by other systems to harvest metadata records from your DSpace.

Enabling OAI-PMH Interface

To enable DSpace's OAI-PMH server, just make sure the `[dspace]/webapps/oai/` web application is available from your Servlet Container (usually Tomcat).

- You can test that it is working by sending a request to: `http://\[full-URL-to-OAI-PMH\]/request?verb=Identify`
- The response should look similar to the response from the DSpace Demo Server: <http://demo.dspace.org/oai/request?verb=Identify>

More information on the OAI-PMH protocol and its usage is available at: <http://www.openarchives.org/pmh/>

OAI-PMH Configuration

Configuration File:	<code>[dspace]/config/modules/oai.cfg</code>
Property:	<code>dspace.oai.url</code>
Example Value:	<code>dspace.oai.url = \${dspace.baseUrl}/oai</code>
Information Note:	This is the OAI-PMH URL for DSpace. By default, it is expected to be <code>\${dspace.baseUrl}/oai</code> , where <code>dspace.baseUrl</code> is defined in your <code>dspace.cfg</code> file.
Property:	<code>didl.maxresponse</code>
Example Value:	<code>didl.maxresponse = 0</code>
Informational Note:	Max response size for DIDL. This is the maximum size in bytes of the files you wish to enclose Base64 encoded in your responses, remember that the base64 encoding process uses a lot of memory. We recommend at most 200000 for answers of 30 records each on a 1 Gigabyte machine. Ultimately this will change to a streaming model and remove this restriction. Also please remember to allocate plenty of memory, at least 512 MB to your Tomcat. Optional: DSpace uses 100 records as the limit for the oai responses. You can alter this by changing the <code>response.max-records</code> configuration below.
Property:	<code>response.max-records</code>
Example Value:	<code>response.max-records = 100</code>
Informational Note:	Maximum number of records to return for OAI-PMH responses. Defaults to 100.

Activating Additional OAI-PMH Crosswalks

DSpace comes with an unqualified DC Crosswalk used in the default OAI-PMH data provider. There are also other Crosswalks bundled with the DSpace distribution which can be activated by editing one or more configuration files. How to do this for each available Crosswalk is described below. The DSpace source includes the following crosswalk plugins available for use with OAI-PMH:

- **met**s - The manifest document from a DSpace METS SIP.
 - **mod**s - MODS metadata, produced by the table-driven MODS dissemination crosswalk.
 - **qdc** - Qualified Dublin Core, produced by the configurable QDC crosswalk. Note that this QDC does *not* include all of the DSpace "dublin core" metadata fields, since the XML standard for QDC is defined for a different set of elements and qualifiers.
- OAI-PMH crosswalks based on Crosswalk Plugins are activated as follows:

1. Uncomment the appropriate `[dspace]/config/oai.properties` of the form: `Crosswalks.plugin_name=org.dspace.app.oai.PluginCrosswalk` (where `plugin_name` is the actual plugin's name, e.g. "met" or "qdc"). These lines are all near the bottom of the file.

- You can also add a brand new custom crosswalk plugin. Just make sure that the crosswalk plugin has a lower-case name (possibly in addition to its upper-case name) in the plugin configuration in `dspace.cfg`. Then add a line similar to above to the `oai.cat.properties` file.
2. Restart your servlet container, e.g. Tomcat, for the change to take effect.
 3. Verify the Crosswalk is activated by accessing a URL such as `http://[full-URL-to-OAI-PMH]/request?verb=ListRecords&metadataPrefix=mets`
 - The response should be an XML document containing METS, similar to the response from the DSpace Demo Server: <http://demo.dspace.org/oai/request?verb=ListRecords&metadataPrefix=mets>

DIDL

By activating the DIDL provider, DSpace items are represented as MPEG-21 DIDL objects. These DIDL objects are XML documents that wrap both the Dublin Core metadata that describes the DSpace item and its actual bitstreams. A bitstream is provided inline in the DIDL object in a base64 encoded manner, and/or by means of a pointer to the bitstream. The data provider exposes DIDL objects via the `metadataPrefix=didl`.

The crosswalk does not deal with special characters and purposely skips dissemination of the `license.txt` file awaiting a better understanding on how to map DSpace rights information to MPEG21-DIDL.

The DIDL Crosswalk can be activated as follows:

1. Uncomment the `didl.maxresponse` configuration in `[dspace]/config/modules/oai.cfg`
2. Uncomment the DIDL Crosswalk entry from the `[dspace]/config/oai.cat.properties` file
3. Restart your servlet container, e.g. Tomcat, for the change to take effect.
4. Verify the Crosswalk is activated by accessing a URL such as `http://[full-URL-to-OAI-PMH]/request?verb=ListRecords&metadataPrefix=didl`
 - The response should be an XML document containing DIDL, similar to the response from the DSpace Demo Server: <http://demo.dspace.org/oai/request?verb=ListRecords&metadataPrefix=mets>

OAI-PMH / OAI-ORE Harvester

This section describes the parameters used in configuring the OAI-ORE / OAI-ORE harvester (for XMLUI only). This harvester can be used to harvest content (bitstreams and metadata) into DSpace from an external OAI-PMH or OAI-ORE server.

Harvesting from another DSpace

If you are harvesting content (bitstreams and metadata) **from** an external DSpace installation via OAI-PMH & OAI-ORE, you first should verify that the external DSpace installation allows for OAI-ORE harvesting.

First, that external DSpace must be running both the OAI-PMH interface and the XMLUI interface to support harvesting content from it via OAI-ORE.

You can verify that OAI-ORE harvesting option is enabled by following these steps:

1. First, check to see if the external DSpace reports that it will support harvesting ORE via the OAI-PMH interface. Send the following request to the DSpace's OAI-PMH interface: `http://[full-URL-to-OAI-PMH]/request?verb=ListRecords&metadataPrefix=ore`
 - The response should be an XML document containing ORE, similar to the response from the DSpace Demo Server: <http://demo.dspace.org/oai/request?verb=ListRecords&metadataPrefix=ore>
2. Next, you can verify that the XMLUI interface supports OAI-ORE (it should, as long as it's a current version of DSpace). First, find a valid Item Handle. Then, send the following request to the DSpace's XMLUI interface: `http://[full-URL-to-XMLUI]/metadata/handle/[item-handle]/ore.xml`
 - The response should be an OAI-ORE (XML) document which describes that specific Item. It should look similar to the response from the DSpace Demo Server: <http://demo.dspace.org/xmlui/metadata/handle/10673/3/ore.xml>

OAI-PMH / OAI-ORE Harvester Configuration

There are many possible configuration options for the OAI harvester. Most of them are technical and therefore omitted from the `dspace.cfg` file itself, using hard-coded defaults instead. However, should you wish to modify those values, including them in `oai.cfg` will override the system defaults.

Configuration File:	<code>[dspace]/config/modules/oai.cfg</code>
Property:	<code>harvester.eperson</code>
Example Value:	<code>harvester.eperson = admin@myu.edu</code>
Informational Note:	The EPerson under whose authorization automatic harvesting will be performed. This field does not have a default value and must be specified in order to use the harvest scheduling system. This will most likely be the DSpace admin account created during installation.
Property:	<code>dspace.oai.url</code>
Example Value:	<code>dspace.oai.url = \${dspace.baseUrl}/oai</code>
Informational Note:	The base url of the OAI-PMH disseminator webapp (i.e. do not include the <code>/request</code> on the end). This is necessary in order to mint URIs for ORE Resource Maps. The default value of <code>\${dspace.baseUrl}/oai</code> will work for a typical installation, but should be changed if appropriate. Please note that <code>dspace.baseUrl</code> is defined in your <code>dspace.cfg</code> configuration file.
Property:	<code>ore.authoritative.source</code>

Example Value:	<code>ore.authoritative.source = oai xmlui</code>
Informational Note:	<p>The webapp responsible for minting the URIs for ORE Resource Maps. If using oai, the <code>dspace.oai.url</code> config value must be set.</p> <ul style="list-style-type: none"> When set to 'oai', all URIs in ORE Resource Maps will be relative to the OAI-PMH URL (configured by <code>dspace.oai.url</code> above) When set to 'xmlui', all URIs in ORE Resource Maps will be relative to the DSpace Base URL (configured by <code>dspace.url</code> in the <code>dspace.cfg</code> file) <p>The URIs generated for ORE ReMs follow the following convention for either setting: <code>http://[base-URL]/metadata/handle/[item-handle]/ore.xml</code></p>
Property:	<code>harvester.autoStart</code>
Example Value:	<code>harvester.autoStart = false</code>
Informational Note:	Determines whether the harvest scheduler process starts up automatically when the XMLUI webapp is redeployed.
Property:	<code>harvester.oai.metadataformats.PluginName</code>
Example Value:	<pre>harvester.oai.metadataformats.PluginName = \ http://www.openarchives.org/OAI/2.0/oai_dc/, Simple Dublin Core</pre>
Informational Note:	<p>This field can be repeated and serves as a link between the metadata formats supported by the local repository and those supported by the remote OAI-PMH provider. It follows the form <code>harvester.oai.metadataformats.PluginName = NamespaceURI,Optional Display Name</code>. The <code>pluginName</code> designates the metadata schemas that the harvester "knows" the local DSpace repository can support. Consequently, the <code>PluginName</code> must correspond to a previously declared ingestion crosswalk. The namespace value is used during negotiation with the remote OAI-PMH provider, matching it against a list returned by the <code>ListMetadataFormats</code> request, and resolving it to whatever <code>metadataPrefix</code> the remote provider has assigned to that namespace. Finally, the optional display name is the string that will be displayed to the user when setting up a collection for harvesting. If omitted, the <code>PluginName:NamespaceURI</code> combo will be displayed instead.</p>
Property:	<code>harvester.oai.oreSerializationFormat.OREPrefix</code>
Example Value:	<pre>harvester.oai.oreSerializationFormat.OREPrefix = \ http://www.w3.org/2005/Atom</pre>
Informational Note:	This field works in much the same way as <code>harvester.oai.metadataformats.PluginName</code> . The <code>OREPrefix</code> must correspond to a declared ingestion crosswalk, while the <code>Namespace</code> must be supported by the target OAI-PMH provider when harvesting content.
Property:	<code>harvester.timePadding</code>
Example Value:	<code>harvester.timePadding = 120</code>
Informational Note:	Amount of time subtracted from the from argument of the PMH request to account for the time taken to negotiate a connection. Measured in seconds. Default value is 120.
Property:	<code>harvester.harvestFrequency</code>
Example Value:	<code>harvester.harvestFrequency = 720</code>
Informational Note:	How frequently the harvest scheduler checks the remote provider for updates. Should always be longer than <i>timePadding</i> . Measured in minutes. Default value is 720.
Property:	<code>harvester.minHeartbeat</code>
Example Value:	<code>harvester.minHeartbeat = 30</code>
Informational Note:	The heartbeat is the frequency at which the harvest scheduler queries the local database to determine if any collections are due for a harvest cycle (based on the <i>harvestFrequency</i>) value. The scheduler is optimized to then sleep until the next collection is actually ready to be harvested. The <i>minHeartbeat</i> and <i>maxHeartbeat</i> are the lower and upper bounds on this timeframe. Measured in seconds. Default value is 30.
Property:	<code>harvester.maxHeartbeat</code>
Example Value:	<code>harvester.maxHeartbeat = 3600</code>
Informational Note:	The heartbeat is the frequency at which the harvest scheduler queries the local database to determine if any collections are due for a harvest cycle (based on the <i>harvestFrequency</i>) value. The scheduler is optimized to then sleep until the next collection is actually ready to be harvested. The <i>minHeartbeat</i> and <i>maxHeartbeat</i> are the lower and upper bounds on this timeframe. Measured in seconds. Default value is 3600 (1 hour).
Property:	<code>harvester.maxThreads</code>
Example Value:	<code>harvester.maxThreads = 3</code>
Informational Note:	How many harvest process threads the scheduler can spool up at once. Default value is 3.
Property:	<code>harvester.threadTimeout</code>

Example Value:	<code>harvester.threadTimeout = 24</code>
Informational Note:	How much time passes before a harvest thread is terminated. The termination process waits for the current item to complete ingest and saves progress made up to that point. Measured in hours. Default value is 24.
Property:	<code>harvester.unknownField</code>
Example Value:	<code>harvester.unknownField = fail add ignore</code>
Informational Note:	You have three (3) choices. When a harvest process completes for a single item and it has been passed through ingestion crosswalks for ORE and its chosen descriptive metadata format, it might end up with DIM values that have not been defined in the local repository. This setting determines what should be done in the case where those DIM values belong to an already declared schema. <i>Fail</i> will terminate the harvesting task and generate an error. Ignore will quietly omit the unknown fields. Add will add the missing field to the local repository's metadata registry. Default value: fail .
Property:	<code>harvester.unknownSchema</code>
Example Value:	<code>harvester.unknownSchema = fail add ignore</code>
Informational Note:	When a harvest process completes for a single item and it has been passed through ingestion crosswalks for ORE and its chosen descriptive metadata format, it might end up with DIM values that have not been defined in the local repository. This setting determines what should be done in the case where those DIM values belong to an unknown schema. Fail will terminate the harvesting task and generate an error. Ignore will quietly omit the unknown fields. Add will add the missing schema to the local repository's metadata registry, using the schema name as the prefix and "unknown" as the namespace. Default value: fail .
Property:	<code>harvester.acceptedHandleServer</code>
Example Value:	<pre>harvester.acceptedHandleServer = \ hdl.handle.net, handle.test.edu</pre>
Informational Note:	A harvest process will attempt to scan the metadata of the incoming items (identifier.uri field, to be exact) to see if it looks like a handle. If so, it matches the pattern against the values of this parameter. If there is a match the new item is assigned the handle from the metadata value instead of minting a new one. Default value: <i>hdl.handle.net</i> .
Property:	<code>harvester.rejectedHandlePrefix</code>
Example Value:	<code>harvester.rejectedHandlePrefix = 123456789, myeduHandle</code>
Informational Note:	Pattern to reject as an invalid handle prefix (known test string, for example) when attempting to find the handle of harvested items. If there is a match with this config parameter, a new handle will be minted instead. Default value: <i>123456789</i> .

Activating / Using the OAI-PMH / OAI-ORE Harvester

For information on activating & using the OAI-PMH / OAI-ORE Harvester to harvest content into your DSpace, see [Harvesting Items from XMLUI via OAI-ORE or OAI-PMH](#)