

## Provider Documentation

This documentation includes the following sections:

- [Overview](#)
- [Provider setup](#)
- [Preparing files for serving](#)
- [Providing files in multiple formats](#)
- [Frequently Asked Questions \(FAQ\)](#)

### Overview

The jOAI data provider allows XML files from a file system to be exposed as items in an OAI data repository and made available for harvesting by others using the OAI-PMH. After pointing the software to one or more file directories, the software monitors the XML files inside, adding, updating or deleting them from the OAI repository as files are added, updated or deleted from the directories. Remote harvesters that monitor the OAI data repository can effectively mirror the files or harvest them as needed. jOAI can provide any XML format as long as the XML in the file is well formed.

The jOAI data provider implements protocol version 2.0. It uses resumption tokens for [flow control](#) in the [ListIdentifiers](#) and [ListRecords](#) responses, supports selective harvesting by [date](#) or [set](#), provides gzip [response compression](#) and other protocol features.

---

### Provider setup

There are five steps necessary to make metadata files available through the jOAI data provider:

1. Install the jOAI software on a system in a servlet container such as [Apache Tomcat](#).

See [INSTALL.txt](#) for installation instructions. If reading this page, most likely this step has been completed.

2. Complete the [Repository Information](#) by clicking 'Edit repository info' in the Repository Information and Administration page and then:

- Enter a repository name (*required*)
- Include an administrators e-mail address (*required*)
- Provide a namespace identifier (*optional but strongly recommended*)
- Provide a description (*optional*)

The **namespace-identifier** is similar to an Internet domain name, for example "dlese.org" or "project.dlese.org." If specified, the namespace identifier is used to compose the OAI Identifier for items in the repository. See the [OAI Identifier Format](#) guidelines for more information.

Leave the **description** blank if not using.

3. Complete the [Metadata Files Configuration](#) in the Metadata Files Configuration page by adding one or more metadata directories to the repository. For each directory:

- Enter an appropriate nickname for the directory of files (*required*)
- Provide the metadata format (metadataPrefix) of the files (*required*)
- Enter the complete directory path to the metadata files (*required*)
- Enter the metadata namespace and schema for the format (*optional but recommended*)

The **directory of files** must contain XML files that conform to the rules described below under [Preparing files for serving](#).

The **metadata format** may be any metadata (or data) format. In the OAI protocol, the format specifier is known as the [metadataPrefix](#). The metadataPrefix may be any combination of [URI unreserved characters](#), such as letters, numbers, underscores and dashes.

Examples:

```
oai_dc      - Dublin Core format
adn         - ADN format
dlese_anno  - DLESE annotation format
dlese_collect - DLESE collection format
news_opps   - DLESE news and opportunities format
```

In general, the **metadata namespace and schema** can be found near the top of an XML file for the given format. If the format is recognized by the software these fields will be filled in automatically.

4. After completing step 3, the software automatically indexes the metadata files, which may take several minutes to complete. Once the files are indexed, the metadata is available for harvesting, for browsing using the OAI protocol via the [Explorer](#) page and for textual searching using the [Search](#) or [Admin search](#) pages. The [Metadata Files Configuration](#) shows information about the status of the files and indexing process. If metadata files are added, modified or deleted at a later time, the software automatically detects these changes and adds, deletes or re-indexes them every 8 hours. The index can also be updated manually at any time from the [Files index administration](#) area.

5. Complete [Sets Configuration](#). This step is *optional*. Define a new set and then:

- Enter a set name (*required*)
- Enter a setSpec (*required*)
- Provide a set description (*optional*)
- Add records to the newly created set by defining which records to include in the set (*required*)

The **set name** is a descriptive name for a group of metadata files that are a subgroup of all metadata files in the repository, for example "DLESE Community Collection".

The **setSpec** is a unique name or label that identifies the subgroup of metadata files; harvesters may use a setSpec to identify and get the correct set of metadata files from providers. A setSpec example is "dcc."

Limit the number files in a set by specifying certain directories, metadata formats or search criteria.

The optional **description** field contains information about the content, purpose, rights or history of the provider. Leave the description field blank if no information is available.

After completing the steps above, metadata files are available for harvesting by others.

This software supports the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), version 2.0. Detailed information about the protocol is outside the scope of this documentation. For background information on the OAI-PMH, please refer to the official [OAI-PMH documentation](#) and to additional information and tutorials available through the [Open Archives Initiative](#).



---

## Preparing files for serving

jOAI monitors each directory of files that is configured in the system and automatically adds, updates or deletes items from the OAI repository as files are added, updated or deleted from the directories. After the initial configuration, the synchronization between the files and the OAI repository occurs automatically every 8 hours or may be [synchronized manually](#) at any time.

To ensure proper operation, files must follow these conventions:

- Each file must be a well-formed XML instance document.
- Each file must contain a single record, which corresponds to a single item in the OAI repository.
- All files within a given directory must be of the same metadata (XML) format.
- Each file name must end with a .xml file extension.

- The file name up to the .xml file extension must indicate a [unique identifier](#) for the record\*. For example, if the file is named `abc-123.xml`, the identifier for the record will be `abc-123` (the identifier is used as the local identifier portion of the [OAI Identifier Format](#) in the OAI protocol). Identifiers in jOAI are *not* case sensitive. \*Note that for files in the `adn`, `dlese_anno`, `dlese_collect` and `news_opps` formats, the file name is not used and instead the identifier must be indicated in the proper location in the file's XML.
- Identifiers must be unique across all files configured in the data provider. It is an error to have two or more files, regardless of format, with the same identifier.
- For network efficiency, the file modification date should change only when the content of the file is modified. Changing the modification date initiates a transfer of the record from the data provider to the harvester.
- To comply with the [XML response format](#) required by the OAI protocol, encoding of the XML must use the UTF-8 representation of Unicode and character references, rather than entity references, must be used.



## Providing files in multiple formats

jOAI can disseminate any given metadata file in multiple formats. For example, a file that resides in the `adn` format can also be disseminated to harvesters in the `oai_dc` format. This is done using metadata format converters. Several metadata format converters come pre-configured in the software as detailed below. New converters can be configured and implemented using an XSL stylesheet or a custom Java class that converts metadata from its native XML format to another XML format. Once a format converter is configured, all files in the native format will be disseminated in either the native or converted format depending on which format is requested by the harvester.

The software comes pre-configured with the following metadata format converters:

Native XML format	Converted XML format
<code>adn</code>	<code>oai_dc</code> , <code>nsdl_dc</code> , <code>briefmeta</code>
<code>dlese_anno</code>	<code>oai_dc</code>
<code>dlese_collect</code>	<code>oai_dc</code>
<code>news_opps</code>	<code>oai_dc</code>

To configure additional metadata format converters, do the following:

1. Create or obtain an XSL stylesheet or Java class that performs the desired format conversion from one XML format to another. The converter takes XML in the native format as its input and must generate XML in the converted format as its output. For Java converters, the class must implement the [XMLFormatConverter](#) Interface.
2. If using an XSL stylesheet to perform the conversion, place it in the "xsl\_files" directory located in the "WEB-INF" directory of the OAI software context. If using a Java class to perform the conversion, place the class binary anywhere within the classpath of the servlet container.
3. Edit the "web.xml" file located in the "WEB-INF" directory and add a context-param element to configure each format converter (see the existing ones for examples). When configuring an XSL converter, the param-name element must start with the string with "xslconverter," followed by additional descriptive text. When configuring a Java class converter, the param-name element must start with the string with "javaconverter," followed by additional descriptive text. Each param-name must be unique; otherwise it will not be recognized.

For the param-value field, supply a string of the form

```
[convertername] | [from format] | [to format]
```

where `convertername` is either the name of an XSL file or a fully qualified Java class and the "to" and "from" formats are metadataPrefixes for the given formats.

For example, an XSL stylesheet named `myDCConverter.xsl` that converts from ADN to Dublin Core, the param-value would be

```
"myDCConverter.xsl|adn|oai_dc" (quotes omitted).
```

For a Java class converter by the full name org.institution.converter.MyDCConverter, the param-value would be

```
"org.institution.converter.MyDCConverter|adn|oai_dc" (quotes omitted).
```

An example of a complete context-param configuration for a format converter looks like the following:

```
<context-param>
  <param-name>xslconverter - adn to oai_dc converter</param-name>
  <param-value>adn-v0.6.50-to-oai_dc.xsl|adn|oai_dc</param-value>
</context-param>
```

4. After configuring the web.xml file and placing the converter in the appropriate location, start or restart the software. The software automatically recognizes the converter and adds the new format to its list of available formats and exposed in response to the OAI [ListMetadataFormats](#) request.

Tip: The format converter module caches the converted metadata to disk for increased performance. These converted files may be accessed locally. Accessed files are cached in the "WEB-INF/repository\_data/converted\_xml\_cache" directory.



---

## Frequently Asked Questions (FAQ)

### If my records reside in a database, can I use jOAI to implement an OAI data provider for my repository?

The jOAI data provider allows XML files from a file system to be exposed as items in an OAI data repository. To expose records that reside in a database, write a routine to export the records to XML files at regular intervals such as once a day or once a week, depending on how often the records change. Then [setup the data provider](#) to monitor the directory or directories where the files are exported to. See also [preparing files for serving](#).

After the initial set of records have been exported from the database, files should be modified or deleted only when the corresponding database record has been updated or deleted. jOAI will monitor the files and provide them to harvesters according to the OAI-PMH.

### Does jOAI support selective or incremental harvesting?

Yes. After performing an initial full harvest of the repository, harvesters may use [datestamps](#) to request only those records that have changed or been deleted since that last time of harvest, which can greatly reduce the number of records transferred over the network over time. The data provider implements deleted records and datestamps in accordance with the OAI-PMH to support selective and incremental harvests.

### If I remove a file, can I add it back at a later time?

When a file is removed from a directory that is being monitored by the data provider, its record will be moved to status deleted, and harvesters will be notified that the record has been deleted the next time they harvest from the data provider. At a later time, if a file is added back with the same unique ID as a deleted record (regardless of the directory), the data provider will replace it with the new one, and its status will no longer be deleted. Harvesters will then receive the new record the next time they harvest from the data provider.

### What happens if I accidentally create two files with the same ID?

When jOAI imports a new or modified file into its index, a check is performed to see if there is an existing record with the same unique ID. If the ID already exists, an error will be reported under 'Indexing Errors' in the [Metadata Files Configuration](#) page, and the file will not be imported into the repository index. To fix the problem the file must either be removed from the directory or a unique ID should be assigned to the file, as described above under [preparing files for serving](#).

### My records have indexing errors when I put them in the data provider. What's wrong?

XML files that contain text that was copied and pasted from tools such as Microsoft Word often contain invalid characters such as dashes or copyright symbols that are improperly encoded. These

'bad characters' can trigger the XML processors in the software to issue an error. Files must contain well-formed XML and should use UTF-8 encoding. Character references, rather than entity references, should also be used for special characters, as required by the OAI protocol [XML response format](#).

### **How many records can the data provider scale to?**

The jOAI data provider is designed for small to medium size data repositories. The software has been tested successfully with repositories up to 300,000 records. The number of records the software can support depends on the amount of memory available to the Java JVM, the speed of the host machine and the size of the individual records in the repository.

### **Can jOAI be configured to run from a virtual host that is served by apache (httpd)?**

In some cases it may be desirable to map jOAI to a virtual host web address and have it served by an apache web server (httpd). For example, a virtual host web address such as <http://oai.somewhere.edu> may be preferred to a standard Tomcat address such as <http://somewhere.edu:8080/oai/>, and functionality available from the apache web server such as consistent web logging or SSL may also be desired.

The [Apache Tomcat Connector](#) (mod\_jk) is used for this purpose by proxying requests made to an apache web server through to Tomcat. Using the connector, a rule may be set up that says, in effect, 'for all requests that come to <http://oai.somewhere.edu>, serve them from the Tomcat context <http://somewhere.edu:8080/oai/>'. Setting up mod\_jk takes some work, which is described in the [Apache Tomcat Connector documentation](#).

### **The baseURL that is shown uses the local machine name, but it should use the domain name for the server instead. How can I change it?**

The base URL that is shown on the [front page](#) and [Repository Information page](#) of jOAI and elsewhere reflects the URL that was entered into the web browser when connecting to the software. For example, if a user accesses jOAI using the web address <http://localhost:8080/oai/>, the baseURL will be shown as 'http://localhost:8080/oai/provider'. If the user connects to the same instance of jOAI using the Internet address <http://myserver.somewhere.edu/oai/>, the baseURL will be shown as 'http://myserver.somewhere.edu/oai/provider'.

