



IMu Documentation

OAI: New IMu Webservice

Document Version 1

EMu Version 4.0



Contents

SECTION 1	What is OAI?	1
	Where to find general information on OAI	2
	Some OAI Terminology and Practice	3
SECTION 2	OAI - a Brief Tour	5
	Interface: the Six verbs	5
	Extra parameters	7
	XML	8
	Flow control	9
SECTION 3	Setting up a KE IMu OAI web service	11
	Configuration	12
	How localisation is done	13
	Where Configuration files are located	14
	File Names	15
	OAI Service	15
	Metadata formats	16
	Configuration File Structure	17
	Simple settings	18
	A list of items	19
	Associative array of items	20
	OAI Service Configuration	21
	Some important parameters	22
	Example	23
	Metadata Configuration	24
	Some important parameters	25
	How field mappings are specified	26
	Special field mappings	27
	Example of a metadata configuration file	28
	Index	29

SECTION 1

What is OAI?

OAI is a web service that returns XML data in response to arguments passed to it. Depending on the passed arguments it is able to return information about the service and its capabilities, as well as record data.

The XML record data can be wrapped in various metadata formats. As a minimum the system will use Dublin Core, however a data source (a repository) may also implement other formats and can wrap the data on request in an alternative format.

OAI has limited query flexibility (confined to data modification dates or to querying preconfigured sets of records) as it is primarily meant for data harvesting rather than generalised querying.

EMu via the IMu Web framework can act as an OAI repository, providing an OAI web service to OAI harvesters.

Where to find general information on OAI

- <http://www.openarchives.org>
- <http://www.openarchives.org/documents/FAQ.html>
- <http://www.openarchives.org/OAI/openarchivesprotocol.html>
- http://en.wikipedia.org/wiki/Open_Archives_Initiative_Protocol_for_Metadata_Harvesting

Some OAI Terminology and Practice

- Data sources are called **repositories**.
- Systems that access repositories are called **harvesters**.
- OAI is designed as a harvesting protocol rather than a generalised search protocol.
- Typically data is harvested by date range (i.e. give me what has changed between these dates).
- Despite not having arbitrary search criteria, a repository can provide data sets which are in effect a pre-defined grouping of records. For example, a repository may define sets such as *The James Cook Images* or *The Fred Nerk Collection*. Harvesters can ask for records from just a specific set.
- OAI allows the use of arbitrary metadata formats rather than a single fixed one. However, as a minimum requirement it must always be able to provide Dublin Core metadata.
- OAI should have functionality for deleted records (<http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm#DeletedRecords>). However this might simply involve the repository declaring when asked that 'I don't maintain info about deleted records'.

SECTION 2

OAI - a Brief Tour

Interface: the Six verbs

Harvesters have six methods (or verbs) to use when communicating with a repository. Each verb, included in a URL request sent to the repository, returns an XML response. Some verbs require or allow additional parameters:

Verb	Description
Identify	Requests a summary of the service. For example: <code>http://caltechcstr.library.caltech.edu/perl/oai2?verb=Identify</code>
ListMetadataFormats	Requests a description of which metadata formats the repository can provide records in. Must at least return Dublin Core but can optionally return others. For example: <code>http://kamome.lib.ynu.ac.jp/dspace-oai/request?verb=ListMetadataFormats</code> In this case the repository provides Dublin Core and something called junii2 (which includes link to schema).
ListSets	Requests a list of sets (i.e. predefined queries). For example: <code>http://genesis2.jpl.nasa.gov/perl/oai2?verb=ListSets</code> In this case the repository has five sets: • Status = Unpublished • Status = Published • Status = In Press • Subject = Presentations • Subject = Publications

Verb	Description
ListIdentifiers	Requests a list of record identifiers. All records in a repository must have a unique identifier (for example, KE's typically uses a mixture of the institution name plus EMu module plus the record's IRN to make that identifier). This verb is asking the system to provide a list of the unique record identifiers. For example: http://genesis2.jpl.nasa.gov/perl/oai2?verb=ListIdentifiers&metadataPrefix=oai_dc
ListRecords	Requests a list of records. For example: http://genesis2.jpl.nasa.gov/perl/oai2?verb=ListRecords&metadataPrefix=oai_dc&set=7375626A656374733D67656E657369732D707562 In this case the list of records is filtered by set.
GetRecord	Requests an individual record. For example: http://genesis2.jpl.nasa.gov/perl/oai2?verb=GetRecord&metadataPrefix=oai_dc&identifier=oai%3AGenericEPrints%3A200701001

Extra parameters

Some verbs require or allow extra parameters. Extra parameters are:

Parameter	Description
from	Optional for ListIdentifiers, ListRecords. For example: from=1998-01-15 This restricts the results to records modified on or after the given date.
until	Optional for ListIdentifiers, ListRecords. For example: until=2008-01-21 This restricts the results to records modified on or before the given date.
set	Optional for ListIdentifiers, ListRecords. For example: set=Cook_Collection Records can be defined as belonging to various sets (defined by the institution). This parameter restricts the results to records that belong to a particular set.
metadataPrefix	Required for GetRecord, ListIdentifiers or ListRecords.
identifier	Required for GetRecord.
resumptionToken	For flow control (page 9).

XML

Returned record XML is wrapped in some minimal OAI XML and the record in its requested metadata is inserted in a metadata section.

For example, an XML response that uses Dublin Core Metadata might look something like:

```
<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>2011-01-27T23:01:20Z</responseDate>
  <request verb="GetRecord" metadataPrefix="oai_dc"
    identifier="oai:jair.org:806">http://jair.fetch.com/oai2.php</req
uest>
  <GetRecord>
  <record>
  <header>
    <identifier>oai:jair.org:</identifier>
    <timestamp>2001-11-01T00:00:00Z</timestamp>
  </header>
  <!-- METADATA SECTION -->
  <metadata>
    <oai_dc:dc
      xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
      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:title>Infinite-Horizon Policy-Gradient
Estimation</dc:title>
      <dc:description>Gradient-based approaches to direct policy
search in
reinforcement learning have received much
recent attention as a means
to solve problems of partial observability
and to avoid some of the
problems associated with policy degradation
in value-function methods.
</dc:description>
      <dc:publisher>Journal Of Artificial Intelligence
Research</dc:publisher>
      <dc:contributor>P. L. Bartlett</dc:contributor>
      <dc:contributor>J. Baxter</dc:contributor>
      <dc:date>2001-11-01 00:00:00</dc:date>
      <dc:type>Text</dc:type>
      <dc:format>application/pdf</dc:format>
      <dc:identifier>doi:10.1613/jair.806</dc:identifier>
      <dc:language>en</dc:language>
      <dc:rights>Copyright held by the AI Access
Foundation</dc:rights>
    </oai_dc:dc>
  </metadata>
  </record>
</GetRecord>
</OAI-PMH>
```

Flow control

Responses that return lists of data (`ListIdentifiers`, `ListRecords`, `ListSets`) may be returned from the repository as an incomplete list because there is too much data to return in one hit. XML returned will be valid but will not have all records. In this case the repository will return a resumption token with the first block of records. The harvester sends another request with the resumption token to the repository. For example:

```
http://AN.OAI.ORG/OAI-  
script?verb=ListIdentifiers&resumptionToken=xxx45abttyz
```

The response to a passed resumption request may also be incomplete, in which case a new resumption token will be received. This process is repeated as many times as required to return the complete set of data.

SECTION 3

Setting up a KE IMu OAI web service

The IMu system can provide an OAI repository for an EMu installation. For the most part setting up a basic IMu/OAI service is a matter of extracting the IMu components and configuring them as required. If the required service uses Dublin Core (or other metadata formats that KE has already implemented for other customers), there should be no real need for any coding to be done.

Configuration

The OAI service consists of two types of components:

1. The service itself.
2. Any metadata formats provided by the service (of which there must be one but may be many).

Configuration of the service and of each metadata format is done separately.

How localisation is done

There are three levels of localisation:

Level	Description
common	KE provides default configuration settings for the OAI service and basic configuration settings for the mandatory oai_dc metadata format. These settings are common to all customers. Typically they comprise a minimum set of field mappings, XML namespaces, etc.
client	Each customer will also have their own customised configurations for their service and metadata requirements. KE sets these up and distributes them with the IMu code for the customer. These override the common settings. Typically they include things like the customer's name, modules, etc.  These client specific settings will be replaced by the KE version of the customer's setup whenever IMu is upgraded by KE. If you edit files here, make sure the changes are sent to KE to prevent them being overwritten during an upgrade.
local	A site specific configuration overrides settings supplied by KE at the common and client levels. Settings here will not be overwritten by a KE upgrade and are solely for the customer to manage and change at will. Typically they include things such as server details, set definitions that may be changed or added to in an ad-hoc way, admin email addresses, etc.

Configuration settings can be placed at any of the three levels. Settings made at the local level will override any set at the client level and the client level settings will override those at the common level.

Where Configuration files are located

Configuration files can be located in three places:

Level	Location
Default settings (common)	~imu/ws/oai/common
Customer specific settings (client)	~imu/ws/oai/client
Site specific settings (local)	~imu/ws/oai/local

Any settings stored in `local` override those stored in `client` and `client` settings override common settings.

File Names

OAI Service

The OAI service configuration is contained in files named `config.xml` placed in any (or all) of the following directories:

`~imu/ws/oai/common`

`~imu/ws/oai/client`

`~imu/ws/oai/local`

Metadata formats

Any metadata settings (e.g. metadata field mappings to EMu fields, etc.) are placed in files typically named after the metadata format.

For example:

- `oai_dc.xml` has Dublin Core field mappings.
- `oai_pa.xml` has Picture Australia field mappings.

These can be placed in any (or all) of the following directories:

`~imu/ws/oai/common`

`~imu/ws/oai/client`

`~imu/ws/oai/local`

Configuration File Structure

The configuration files are XML and can represent various settings, including:

- Simple setting=value type items
- Lists of values (e.g. a list of field mappings)
- Associative arrays

Simple settings

Simple settings are just name=value and are represented as:

```
<earliestDatestamp>1970-01-01</earliestDatestamp>
```

This translates as the earliestDatestamp is 1970-01-01

A list of items

A simple list of items is represented as:

```
<EMuModules type="array">
  <item>ecatalogue</item>
  <item>eparties</item>
  <item>emultimedia</item>
  <item>enarratives</item>
</EMuModules>
```

This indicates that four EMu modules can participate in the service and they are:

- ecatalogue
- eparties
- emultimedia
- enarratives

The parameter is of type "array" and has one or more items in it.

Associative array of items

Where items in a list need to be named to distinguish them from each other, an attribute called key is used.

For example:

```
<metadataHandlers type="array">
  <item key="oai_dc" type="array">
    <item key="description">default Dublin Core Handler</item>
    <item key="configuration">oai_dc.xml</item>
  </item>
  <item key="oai_pa" type="array">
    <item key="description">default Picture Australia
Handler</item>
    <item key="configuration">oai_pa.xml</item>
  </item>
  <item key="umbl" type="array">
    <item key="description">UMBL mixed DC and EAD
Handler</item>
    <item key="configuration">umbl.xml</item>
  </item>
</metadataHandlers>
```

This XML indicates that there is a set of three metadatahandlers, with the names oai_dc, oai_pa and umbl. Each of these handlers has two properties, description and configuration.

OAI Service Configuration

The root element of the main service OAI configuration XML should be named EMuOaiConfig. As it contains a list of values it has an attribute to say it contains a number of items (an array):

For example:

```
<EMuOaiConfig type="array">  
  ...  
  ...  
</EMuOaiConfig>
```

Some important parameters

Parameter	Description
repositoryName	Displayed on the identify response. For example: The Museum Of Nature's OAI Repository
host	The host of the EMu IMu Server.
port	The port the EMu IMu server is listening on.
identifierScheme	Used to build unique resource identifiers.
identifierNamespace	Used to build unique resource identifiers.
identifierLocalPrefix	Used to build unique resource identifiers.
adminEmail	Email of the person who administers the service.
granularity	Can you search to nearest day or nearest second?
earliestDatestamp	What is the earliest 'from value' that can be requested?
biteSize	The maximum number of records to be returned before a resumption token gets sent and the response completed. Used to throttle the amount of data sent in each transaction.
metadataHandlers (array)	A list of metadata types the service will provide.  Each will require their own configuration file.
knownSets type (array)	A list of sets (and the criteria used to define them) that the service provides.
xslStylesheet	To allow human friendly display of the OAI responses you can specify an XSL stylesheet that will display the response in a styled way when the response is returned to a web browser.
EMuModules type (array)	A list of EMu modules that the service will allow direct access to as records.
deletedRecord	Should be set to no. The IMu OAI system does not support deleted record functionality at the moment but this parameter is provided as such support may be added in the future.

Example

```
<EMuOaiConfig type='array'>
  <repositoryName>Museum of Quirky Stuff OAI
Repository</repositoryName>
  <identifierScheme>oai</identifierScheme>
  <identifierNamespace>quirkymuseum.edu</identifierNamespace>
  <identifierLocalPrefix>EMu.eQuirkyStuff</identifierLocalPrefix>

  <adminEmail>fred@quirkymuseum.edu</adminEmail>
  <granularity>day</granularity>
  <earliestDatestamp>1970-01-01</earliestDatestamp>
  <deletedRecord>no</deletedRecord>
  <biteSize>50</biteSize>

  <metadataHandlers type='array'>
    <item key='oai_dc' type='array'>
      <item key="description">default Dublin Core Handler</item>
      <item key="configuration">oai_dc.xml</item>
    </item>
  </metadataHandlers>

  <knownSets type='array'>
    <item key="multimedia" type='array'>
      <item key="name">multimedia</item>
      <item key="description">All Multimedia Records</item>
      <item key="EMuModule">emultimedia</item>
    </item>
  </knownSets>

  <xslStylesheet>./common/oai2.xsl</xslStylesheet>

  <EMuModules type='array'>
    <item>ecatalogue</item>
    <item>eparties</item>
    <item>emultimedia</item>
    <item>enarratives</item>
  </EMuModules>
</EMuOaiConfig>
```

Metadata Configuration

Each metadata format provided by the service requires a separate configuration file.

The root element of the metadata configuration XML should be named `EMuOaiMetadataHandler`. As it contains a list of values it has an attribute to say it contains a number of items (an array).

For example:

```
<EMuOaiMetadataHandler type="array">
  ...
  ...
</EMuOaiMetadataHandler>
```

Some important parameters

Parameter	Description
namespaces (array)	A list of namespaces that will be declared in the responses.
holdingElement	The root element of any metadata.
holdingElementNamespaces	namespace declarations that need to be declared in the metadata root element.
fieldMappings (array)	A list of mappings that define how the metadata fields are to be assembled.

How field mappings are specified

Each field mapping consists of:

- The metadata field name
-AND-
- A list of EMu modules and fields that contain that metadata.

A default mapping can be made which is used if no specific Module>EMu Field>Metadata Field is defined.

For example:

```
<item key="dc:title" type="array">
  <map key="ecatalogue">EADUnitTitle</map>
  <map key="emultimedia">MulTitle</map>
  <map key="default">SummaryData</map>
</item>
```

This specifies that the dc:title metadata field comes from the EMu field EADUnitTitle for Catalogue records, from the MulTitle field for Multimedia records and from SummaryData for records of any other type.

Special field mappings

In addition to field data coming directly from EMu fields, it is also possible to specify that returned values are either *hard-coded* to always return a fixed value or *templated* to return a mixture of hard-coded text plus data from various EMu fields.

For example, a hard-coded mapping:

```
<item key="dodgy" type="array">
  <map key="default" modifier="static">THIS METADATA FORMAT IS AN
  EXAMPLE - configure as required!</map>
</item>
```

This specifies that the "dodgy" metadata field will always be returned for records from every module and will always have the value THIS METADATA FORMAT IS AN EXAMPLE - configure as required!

A templated mapping:

```
<item key="multimediaLink" type="array">
  <map key="ecatalogue"
    modifier="dynamic">http://someserver?irn=[MulMultiMediaRef_tab.ir
n]&thumb=no</map>
</item>
```

This specifies that the multimedia metadata field will be returned for Catalogue records only and will be made up of a mixture of hard-coded text plus the value of the EMu field:

"MulMultiMediaRef_tab->emultimedia->irn" substituted into it

So for actual records it might appear as:

```
http://someserver?irn=123&thumb=no
...
http://someserver?irn=9856&thumb=no
...
http://someserver?irn=325002&thumb=no
etc
```

For example, using the earlier mapping:

```
<item key="multimediaLink" type="array">
  <map key="ecatalogue"
    modifier="dynamic">http://someserver?irn=[MulMultiMediaRef_tab.irn]&a
mp;thumb=no</map>
</item>
```

and assuming that the Catalogue record being harvested has a value of 98765 in its *MulMultiMediaRef_tab* field, the OAI system would create a value for the *multimediaLink* field of:

```
http://someserver?irn=98765&thumb=no
```

Example of a metadata configuration file

```
<EMuOaiMetadataHandler type="array">
  <namespaces type="array">

    <namespace>http://www.openarchives.org/OAI/2.0/oai_dc/</namespace>

    <namespace>http://www.openarchives.org/OAI/2.0/oai_dc.xsd</namespa
    ce>

    <namespace>http://www.openarchives.org/OAI/2.0/oai_dc/</namespace>
    </namespaces>

    <holdingElement>oai_dc:dc</holdingElement>

    <holdingElementNamespaces>
      xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:dcterms="http://purl.org/dc/terms/"
      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"
    </holdingElementNamespaces>

    <fieldMappings type="array">
      <item key="dc:title" type="array">
        <map key="default">SummaryData</map>
        <map key="ecatalogue">EADUnitTitle</map>
        <map key="emultimedia">MulTitle</map>
      </item>
      <item key="dc:creator" type="array">
        <map key="ecatalogue">CreCreatorRef_tab.SummaryData</map>
        <map key="emultimedia">MulCreator_tab</map>
      </item>
      <item key="dc:subject" type="array">
        <map key="ecatalogue">TitObjectCategory</map>
        <map key="emultimedia">DetSubject_tab</map>
      </item>
      <item key="dc:description" type="array">
        <map key="default">ExtendedData</map>
        <map key="ecatalogue">EADScopeAndContent</map>
        <map key="emultimedia">MulDescription</map>
      </item>
      <item key="dc:publisher" type="array">
        <map key="emultimedia">DetPublisher</map>
      </item>
      <item key="dc:contributor" type="array">
        <map key="emultimedia">DetContributor_tab</map>
      </item>
      ...
      ...
      ...
    </fieldMappings>
  </EMuOaiMetadataHandler>
```


Index

A

A list of items • 12

Associative array of items • 13

C

Configuration • 9

Configuration File Structure • 12

E

Example • 16

Example of a metadata configuration file • 19

Extra parameters • 5

F

File Names • 11

Flow control • 5, 7

H

How field mappings are specified • 17

How localisation is done • 9

I

Interface

the Six verbs • 3

M

Metadata Configuration • 17

Metadata formats • 11

O

OAI - a Brief Tour • 3

OAI Service • 11

OAI Service Configuration • 14

S

Setting up a KE IMu OAI web service • 9

Simple settings • 12

Some important parameters • 14, 17

Some OAI Terminology and Practice • 2

Special field mappings • 18

W

What is OAI? • 1

Where Configuration files are located • 10

Where to find general information on OAI • 1

X

XML • 6