ECP-2006-DILI-510003

TELplus

OAI-PMH implementation and tools guidelines

Deliverable number	*D-2.1*
Dissemination level	*Public*
Delivery date	*31 May 2008*
Status	*Final v1.1*
Author(s)	*Diogo Reis(IST), Nuno Freire(BNP)*

e*Content*plus

This project is funded under the *e*Content*plus* programme[1],
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

[1] OJ L 79, 24.3.2005, p. 1.

Contents

1 Overview

The Open Archive Initiative (OAI) is a coordinated activity dedicated to solving problems of digital library interoperability by defining simple protocols and standards.

The first result of this initiative was the Santa Fe Convention [1], which resulted from a meeting held in October 1999. Its motivations were: the fast growth of the Internet and its adoption by scholars as a medium of choice for sharing results; slow traditional publishing model for the scholarly fields rapidity of advances; transfer of rights from author to publisher blocking dissemination of results; peer-review sometimes working as a barrier to new ideas, articles from prestigious institutions were favoured; subscription prices becoming too high for libraries. The initial OAI target repositories were e-print archives, where research papers submitted by authors are stored.

After considering the complexity of the problem, the Santa Fe recommendations to interoperability were restricted to the level of metadata harvesting. The mechanisms for establishing this interoperability were three-fold:

1. the definition of a set of simple metadata elements - the Open Archives Metadata Set (OAMS) , which could be expressed using the Dublin Core Element Set, with qualifications in some elements, to enable document discovery among archives;
2. use XML as a common syntax for representation and transport;
3. definition of a common protocol - the Open Archives Dienst Subset to enable extraction of OAMS and archive-specific metadata from participating archives.

A modified version of the Dienst protocol [17] was used to prove the concept. This was the basis for a new protocol, the Protocol for Metadata Harvesting (OAI-PMH). The purpose of the OAI-PMH protocol is to transfer metadata from a source archive to a destination archive.

The first version of OAI-PMH was released in January 2001. The current version is 2.0, which resulted from a major revision of the protocol, and was released in June 2006 [2].

2 The OAI-PMH protocol

OAI-PMH "provides an application-independent interoperability framework based on metadata harvesting." [1]. OAI-PMH is based on a client-server architecture and uses XML over HTTP.

An overall picture of what is OAI-PMH is shown in Fig. 1. In simple terms, data to be made available by entities named Data Providers is made accessible trough OAI-PMH servers, seen as Repositories. That data can be collected by OAI-PMH clients, named Harvesters, which are operated by entities called Service Providers (those entities are named "services" because it is assumed that they will reuse the data harvested to provide some kind of service).

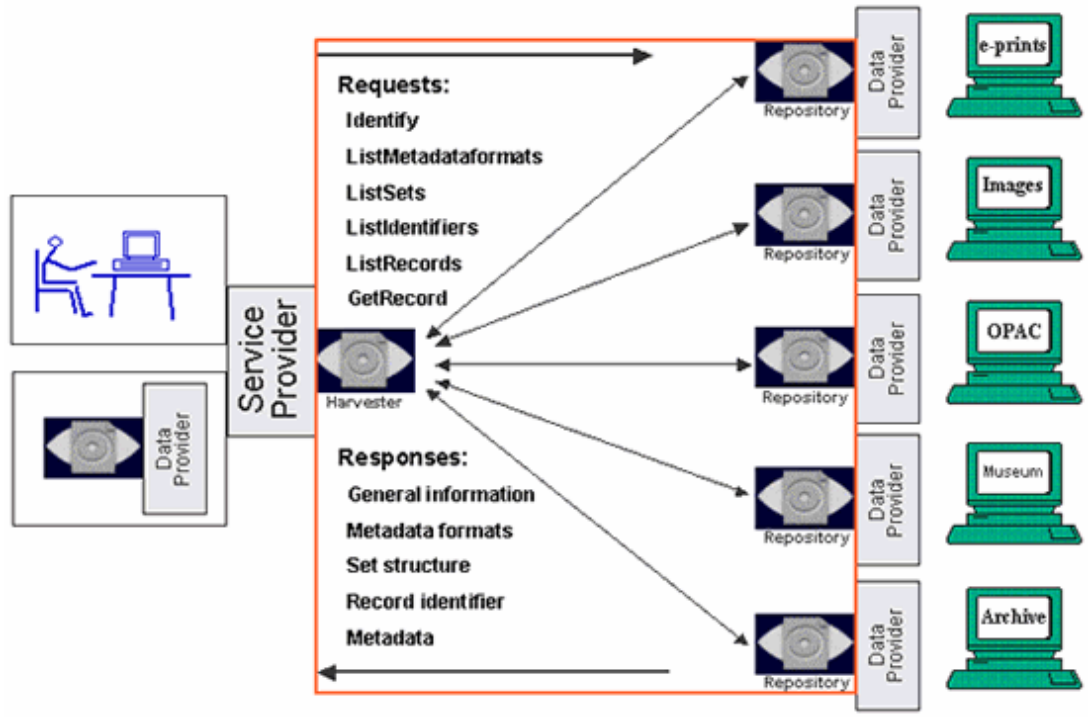

Fig. 1. OAI-PMH: overview and structure model[1]

2.1 OAI-PMH Characteristics

OAI-PMH doesn't support searching, because the purpose is to transfer defined sets of metadata between repositories. The services provided over the data are outside the scope of the protocol. This also means that it's scalable, because all the data needed can be stored in a central repository and searches or processing can be done locally.

There are two types of harvesting criteria that may be combined in an OAI-PMH request: sets and datestamps. The data made available in a repository can be grouped in sets. A repository must have at least one set, but data providers might decide to publish several sets (those sets

[1] Image taken from <http://www.oaforum.org/tutorial/image/structure-model.gif>

can be disjoints or overlap, i.e., a same record can belong to more than one set). The sets can be structured also in multiple levels (so sets can have sub-sets). When harvesting using set selection, all records from that set and descendant sets are given in the response Harvesting using datestamp selection can be used to request only the records within a set which were created, deleted, or modified in a specified date range.

Requests in OAI-PMH are submitted using either the HTTP GET or POST methods. Responses are always XML documents encoded in UTF-8, conforming to an XML Schema, and have a description of the request being responded. Dublin Core [4] is a mandatory format, the least common denominator of interoperability, but a repository can also provide the records in other metadata formats.

An important feature of OAI-PMH is it's simplicity: requests are simply one GET or POST request, responses are given in XML and the protocol is asynchronous (the only information the client needs to keep is the resumption token for large sets of records).

2.2 The main concepts of OAI-PMH

2.2.1 Participant Types

As intended in the Santa Fe Convention, two participant types were defined: Data Providers, which are entities that support the OAI-PMH as a means of exposing metadata; and Service Providers, which use the harvested metadata to build value-added services.

Some systems may act as both.

2.2.1.1 Data Providers

Data Providers refer to entities who possess metadata and are willing to share this with others. These entities provide free access to metadata, and may offer free access to full texts or other resources (but that will be out of the scope of the protocol). One of the goals of OAI-PMH is to provide an easy way to implement a low barrier solution for Data Providers.

2.2.1.2 Service Providers

Service Providers are entities who harvest data from Data Providers in order to provide higher-level services to users. Typical services are searching, browsing, etc. An interesting scenario is when Service Providers can process the data and use it to provide services as Data Providers (such as, for example, harvesting all the records from a group of libraries and provide them again regrouped in new sets mixing records from more than one source).

2.2.2 OAI-PMH Software

2.2.2.1 Harvester

A harvester is a client application that sends OAI-PMH requests. It is operated by a Service Provider as a means of collecting metadata from repositories. It is the mechanism Service Providers have to retrieve data from Data Providers.

2.2.2.2 Repository

A repository is a server accessible in a network, able to process OAI-PMH requests. It is managed by a Data Provider to expose metadata with the purpose of being harvested. To explain what is the Repository there are three concepts that must be described as well:

- *resource* – what the metadata refers to, the nature of which is outside the scope of the OAI-PMH.
- *item* – a constituent of a repository from which metadata about a resource can be disseminated.
- *record* – metadata from an item, in a specific metadata format, encoded in XML.

2.2.2.3 Deployments examples

It is possible to see in Fig. 2 a network of Systems supporting OAI-PMH. There are several Data Providers and Service Providers. Each Service Provider harvests data from several Data Providers.

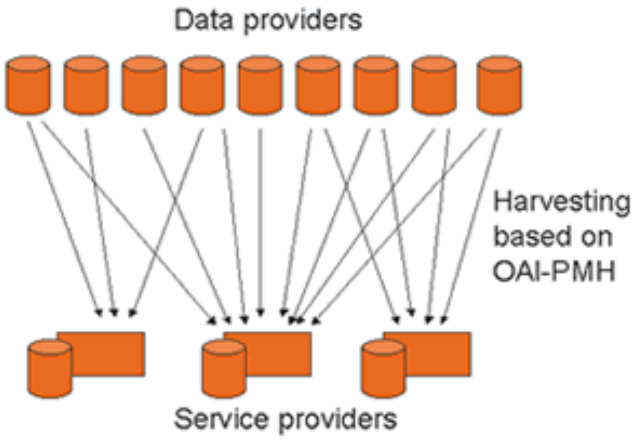

Fig. 2. Multiple Service Providers

A more complex example can be seen in Fig. 3, a network of Systems supporting OAI-PMH. There are several Data Providers for each Service Provider, and also an aggregator that harvests several Data Providers, which can in turn feed Service Providers.

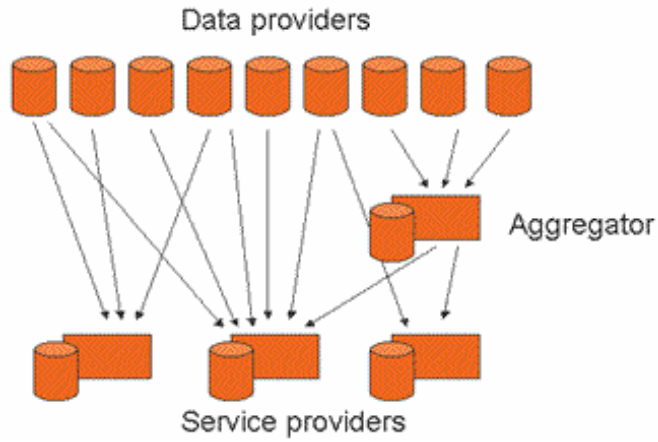

Fig. 3. Aggregators

2.2.3 Metadata Formats

OAI-PMH makes use of the metadata of resources (ex: a PDF file), and not resources themselves. Below the resources are items, the most abstract entity in OAI-PMH. An item has a unique identifier and is the entry point for all the resource metadata. Below the item are records. Records have unique identifiers and contain the metadata in a specific format in XML: MARC, Dublin Core, Qualified Dublin Core, MODS, METS, The European Library application profile for objects (TEL-AP), etc.

2.2.4 Identifiers

An identifier in OAI-PMH must be unique so it can unambiguously identify an item within a repository. The unique identifier maps to an item, and all possible records available from a single item share the same unique identifier.

The format of the unique identifier must correspond to that of the URI (Uniform Resource Identifier) syntax. The scheme component of the unique identifiers must not correspond to that of a recognized URI scheme unless the identifiers conform to that scheme.

Unique identifiers play two roles in the protocol: in the response for the verbs ListIdentifiers and ListRecords to identify each record; and to request a specific record in the request GetRecord.

2.2.5 Repository Sets

Sets are an OAI-PMH mechanism to allow for harvesting of sub-collections, whose semantics are defined outside of the protocol. Sets are defined by conventions established between Data and Service Providers, or just by the Data Provider. They may be several logic aggregations, like: Journal issues, Institutional Repositories, EPrint Archives and Collections with some kind of associated semantics.

In The European Library an OAI Set will be available in the portal as a collection. As such, it will have a collection description in the portal.

2.2.6 Deleted Records

A deleted record means that the record was once in the repository but it is no longer available. There are three types of support for deleted records: no support, persistent and transient. No support means that the repository is not able to keep track of the records that have been deleted, and therefore does not support this feature. Persistent means that when a record is deleted in the repository, it is always possible to see the status of the deleted record at any time and, as such, the repository must keep track of all the deleted records. If the support is transient that implies that the repository only keeps track of deleted records for a period of time, therefore the list of deletions may not be consistent and the deleted status of the record may not be accurate. If a repository keeps track of deletions, the datestamp of the deleted records must represent the deletion date and time.

It is important that repositories have persistent support for deleted records so that OAI-PMH clients may discover deletions through incremental harvesting. If deleted records are not supported, a periodical full harvest is needed to keep the harvested data on the client consistent with the data on the repository. Even though this periodic full harvest makes the harvesting process less efficient, it is not problematic. Difficulties in performing periodic full harvests may be found only in those repositories that hold several million records.

2.3 OAI-PMH Requests and Resumption Token

As described previously, all requests are HTTP GET or POST requests, and all responses, being errors or regular responses, are given in XML. The following sections will describe the six request types available in OAI-PMH.

2.3.1 Identify

The server responds with information about the Repository. Some of that information is required, namely: its name, the base URL, OAI-PMH protocol version, earliest datestamp available, support type for deleted records, the administrator email and time granularity.

There are two granularities for the time: “complete date” (YYYY-MM-DD) and “complete date plust hours, minutes and seconds” (YYYY-MM-DDThh:mm:ssZ), encoded using ISO8601. Every repository must support the “complete date” granularity. Within The European Library, only “complete date” granularity is necessary.

2.3.2 ListMetadataFormats

List the available metadata formats available from the Repository. It is also possible to ask for the metadata formats available for an item giving its identifier as a parameter.

2.3.3 ListSets

This verb is used to get the available sets from the Repository. As an optional argument it is possible to use the Resumption Token for flow control, which will be described in section 2.3.6.

2.3.4 ListIdentifiers and ListRecords

This requests are very similar, both return lists of record identifiers. The difference between them is that the ListRecords response also includes the Records in the metadata format passed as a parameter. It is possible to restrain the list returned by using the arguments from and until, which limit the answer to Records updated after a given time and before a given time respectively (the time format was already mentioned in 2.3.1). It is also possible to limit the Records to a given set by using the setSpec parameter.

If the Repository keeps information about deleted records, the response also includes the records that were deleted, showing in the attribute status of the header the value “deleted”.

As with the verb ListSets, it is possible to use a Resumption Token for flow control, described in section 2.3.6.

2.3.5 GetRecord

Retrieves a Record from the Repository giving the identifier and the metadata format in which it should be returned (metadataPrefix parameter). If the Repository stores the information of deleted records, it is also possible to track if the record was deleted, showing in the attribute status of the header the value “deleted”.

2.3.6 Resumption Token

The verbs ListSets, ListIdentifiers and ListRecords return a list of discrete entities. When those lists are too large to be sent in a single response it may be practical to partition the list and retrieve it in a series of requests and responses. The mechanism OAI-PMH has to deal with this is the use of resumption tokens. When a request can’t be answered in a single response, a list is sent in the response along with a resumption token, which can be used in the subsequent request to get the next partition of the list. The format of the resumption token is not defined by the OAI-PMH and therefore it should be considered opaque by the harvester.

There are two optional attributes in the resumption token: the date of expiration (when the token ceases to be valid) and the complete size of the list requested and the cursor (number of elements of the list returned so far, starting at 0).

2.3.7 OAI-PMH Request Examples

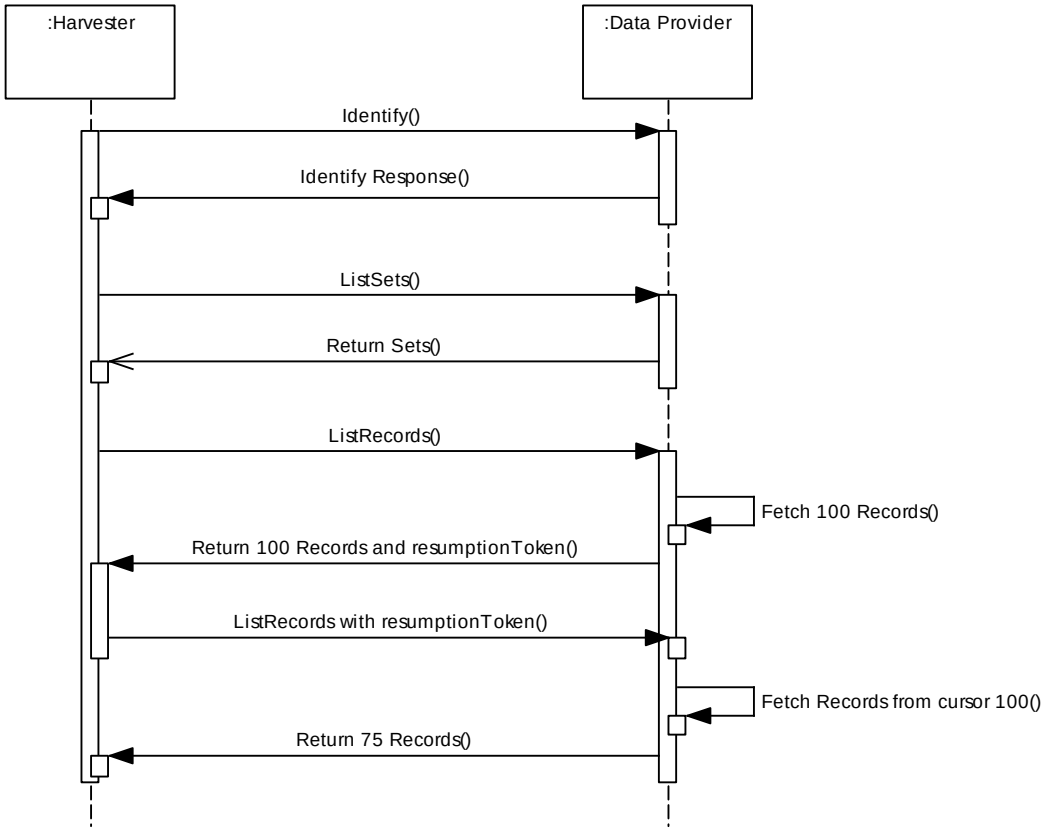

Fig. 4. Example of harvesting a Data Provider

An example of harvesting a Repository having a set with 175 Records is shown in Fig. 4. The first step for the Harvester is to get information on the Repository of the Data Provider with the verb Identify. After that it retrieves the available sets on the Repository with ListSets. Selecting one of the sets, two requests are made. The first is a ListRecords which returns the first 100 Records and a Resumption Token. That Resumption Token is used in the next request to get the remaining records starting from the last one received. Since the set only has 175 Records, the last 75 Records are returned ending the harvesting process.

An example taken from the OAI 2.0 specification [1]:

Request: List the records expressed in the oai_rfc1807 metadata format, that have been added or modified since January 15, 1998 in the hep subset of the physics:

```
http://an.oa.org/OAI-script?verb=ListRecords&from=1998-01-
15&set=physics:hep&metadataPrefix=oai_rfc1807
```

Response: Two records are returned:

```
<?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>2002-06-01T19:20:30Z</responseDate>
 <request verb="ListRecords" from="1998-01-15"
```

```
          set="physics:hep"
          metadataPrefix="oai_rfc1807">
          http://an.oa.org/OAI-script</request>
 <ListRecords>
  <record>
    <header>
      <identifier>oai:arXiv.org:hep-th/9901001</identifier>
      <datestamp>1999-12-25</datestamp>
      <setSpec>physics:hep</setSpec>
      <setSpec>math</setSpec>
    </header>
    <metadata>
     <rfc1807 xmlns=
        "http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1807.txt"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation=
        "http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1807.txt
         http://www.openarchives.org/OAI/1.1/rfc1807.xsd">
         <bib-version>v2</bib-version>
         <id>hep-th/9901001</id>
         <entry>January 1, 1999</entry>
         <title>Investigations of Radioactivity</title>
         <author>Ernest Rutherford</author>
         <date>March 30, 1999</date>
     </rfc1807>
    </metadata>
    <about>
      <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:publisher>Los Alamos arXiv</dc:publisher>
        <dc:rights>Metadata may be used without restrictions as long as
           the oai identifier remains attached to it.</dc:rights>
      </oai_dc:dc>
    </about>
  </record>
  <record>
    <header status="deleted">
      <identifier>oai:arXiv.org:hep-th/9901007</identifier>
      <datestamp>1999-12-21</datestamp>
    </header>
  </record>
 </ListRecords>
</OAI-PMH>
```

2.3.8 Technical documentation

For more technical information on OAI-PMH the following references are recommended:

- The Open Archives Forum online tutorial [5].
- OAI-PMH specification [1].

3 OAI-PMH implementation in libraries

In this section all the steps required to implement OAI-PMH in libraries will be discussed. First, requirements to be a Data Provider are presented, then three common scenarios of implementations: a Library Management System module provided by a vendor; an In-house OAI-PMH Server development, using existing tools; and a Standalone OAI Server providing a full solution. Then, the metadata crosswalks that are handled by the OAI-PMH Server are examined followed by how to test the implementation and finally other implementation tasks.

3.1 Requirements to be a Data Provider

The characteristics to be a Data Provider are:

1. Have a source of metadata, from human or automated resource catalogues.
2. Create metadata mappings (crosswalks from native formats to DC or other formats).
3. Have a Server, which is handled by the OAI software.
4. Datestamps associated to Records, which indicates when the item was last changed (handled by the OAI software).
5. Flow control with resumption tokens (handled by the OAI software).

Some desired features, which are not mandatory, are:

1. Deletions, indicating if the item has been deleted and should be removed (handled by the OAI software).
2. Unique identifiers to identify each item across Repositories, if there is no such information a unique identifier can be created, but updates on the Repository have to be carefully monitored.
3. Organization of records in sets.

3.2 Typical scenarios of OAI-PMH implementation in libraries

3.2.1 Library Management System OAI-module provided by the vendor

3.2.1.1 Description

This solution is adequate for Libraries that already have a Library Management System (LMS) or are in the process of acquiring one, whose vendor also provides an OAI-PMH Server Module (see Fig. 5).

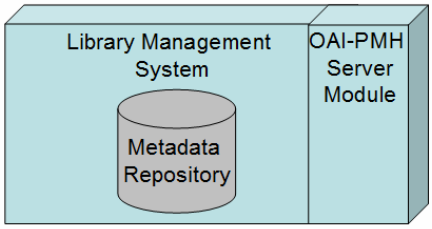

Fig. 5. Library Management System with OAI-PMH Server Module

To obtain an OAI-PMH Server Module there are several steps that must be taken. First it is needed to inquire the vendor for the availability of the module, as many vendors don't have an OAI module for their products. Also make sure that the OAI module supports metadata crosswalks, required for sharing data according to the The European Library application profile.

In this scenario the Library must develop the metadata crosswalk specifications for their data format into simple DC and TEL-AP, if there are not already implemented in the OAI-PMH module. The vendor will implement them on the OAI module.

The advantages of this approach are that the data available by OAI-PMH is always up to date, because the OAI-PMH module is tightly integrated with the LMS being used. No in-house developers are needed since there is no need to adapt the system as the module is specific to the LMS and there is a low maintenance cost.

The disadvantages are that not all vendors offer an OAI module for their products, the costs of the software may be significant depending on the LMS and there is less flexibility in defining metadata crosswalks (in some cases) because each LMS may have their pre-defined transformations and only the software vendor may adapt the crosswalks .

3.2.1.2 Tools

There are several Library Management Systems available, and a simple Web search can be useful to find out the existing ones and compare them. Because it's difficult to perform an exhaustive analysis on all the tools available, they are not going to be mentioned here as it is beyond the scope of this document.

3.2.2 In-house OAI-PMH Server development

3.2.2.1 Description

If a Library has developers available for the task of implementing and configuring the OAI-PMH Server, this may be the best option. There are tools available to provide the infrastructure, which can be very useful to shorten the development time. This scenario is particularly relevant for Libraries that have a Metadata Repository of difficult access that needs a non-standard implementation. A typical architecture for this approach can be seen in Fig. 6.

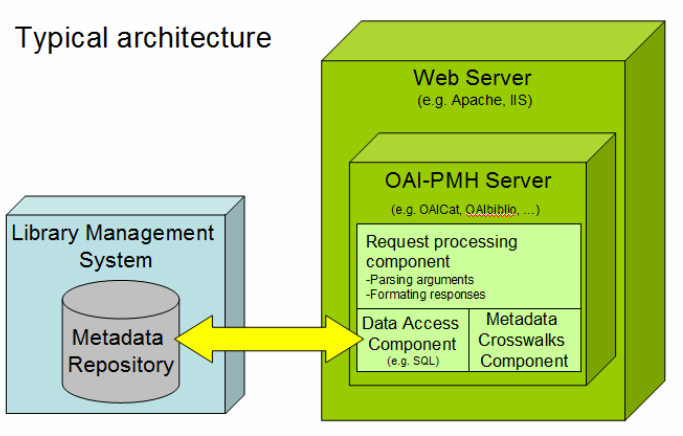

Fig. 6. In-house OAI-PMH Server

To develop an OAI-PMH Server the IT team must first check if it is possible to access the metadata repository of the LMS and also which programming languages can be used. Then confirm that the LMS has unique identifiers and date stamps on the records. An open source OAI-PMH implementation should be chosen, taking in consideration the programming language to use, based also on the skills of the software developers. It is also possible to create an OAI-PMH implementation, but since there are good tools working and tested, it is unnecessary. Finally there must be developed metadata crosswalk specifications for the Library's format and then test the server.

Detailing the components that are needed for an In-house OAI-PMH Server (the green components on the right of Fig. 6):

- Web Server – use an existing one that works with the tool used to implement OAI-PMH (like Apache, Tomcat, jetty, IIS, etc.).
- OAI-PMH Server – the tool actually used to process the OAI-PMH workflow. It is possible to implement this server, but it is not advised because it would be duplicating effort, given the number of implementations that exist for several programming languages.
- Request processing component – it is the part of the OAI-PMH Server that parses the OAI-PMH requests and builds the respective responses (typically, this component is completely handled by the chosen tool).
- Data Access Component – this is the component that actually manages the data (listing sets, retrieving records, etc.). Most tools should have an infrastructure available to simplify the process of accessing the data. Some tools provide implementations for accessing records in different databases or XML files.
- Metadata Crosswalks Component – The component that converts from one metadata format to another and manages available metadata formats for records and sets. Although most OAI-PMH tools provide support for handling the conversions, the actual transformation should be provided or implemented by the Library.

The advantages of this approach are: the data available by OAI-PMH is always up to date; it requires only free open source software, which in turn lowers the total cost of implementation; it has low maintenance costs; and is a good option for Libraries using an in-house developed LMS or repository because the IT team has a deep knowledge of how the system works and how they can adapt it.

The shortcoming are that not all libraries have personnel with the required knowledge, outsourcing is possible but it may be hard to find companies with OAI experience. This may be the most time consuming scenario if the OAI-PMH Server is not flexible enough to perform the necessary interfaces to access the data, although in some cases the software development can be done in just 2-5 days work.

3.2.2.2 Tools

Among the many tools available, there are general OAI-PMH implementations in several languages to plug-in into a Library Management System. The following list contains examples of such tools available in several programming languages:

- Java: The OCLC OAICat project is a Java Servlet web application providing a repository framework that conforms to the OAI-PMH v2 [6].
- PHP: OAIbiblio is a data provider implementation of the OAI-PMH, version 2.0. This toolkit can be easily customized to communicate with an already existing, multi-table MySQL database [7].
- PERL: Celestial is an OAI aggregator/cache application that imports OAI metadata from version 1.0,1.1,2.0 OAI-compliant repositories, and re-exposes that metadata through either an aggregated or per-repository OAI-compliant 2.0 interface. Celestial requires oai-perl v2, MySQL, Perl 5.6.x and a CGI-capable web server [8].
- Ruby: ruby-oai includes a client library, a server/provider library and a interactive harvesting shell [9].
- Python: The pyoai package enables high-level access to an OAI-PMH Metadata Repository and also implements a framework for quickly creating OAI-PMH compliant servers [10].
- More servers at [11].

3.2.3 Standalone OAI Server

3.2.3.1 Description

This scenario, depicted in Fig. 7, involves adding an external OAI-PMH server that runs without an online connection to the LMS repository. In this scenario the metadata is exported from the LMS by its own export functionality, and then imported into the OAI-PMH server.

This approach is appropriate for Libraries without a software development team. This scenario is the simplest, as there is little effort required to set up the OAI server. The focus will be in setting up the metadata export/import process.

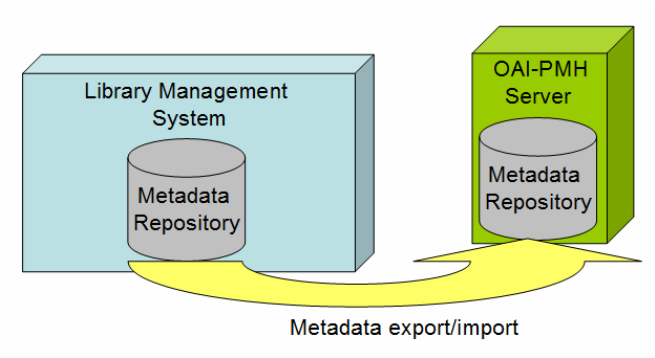

Fig. 7. Standalone OAI Server scenario

To use a standalone OAI Server, the first task is to choose an open source standalone OAI-PMH server, then install and configure it. Afterwards it is necessary to develop metadata crosswalk specifications for the Library's format. Then follows the implementation of a procedure in the library to transfer the metadata records from the LMS to the OAI-PMH server. Finally, test the server.

To implement a procedure in the library to transfer the metadata records from the LMS to the OAI-PMH server:

- Export metadata records from the LMS (Typically in ISO2709 or MarcXchange; if possible, just the records that changed since the last update).
- Execute the metadata conversion by generating XML records in TEL-AP and DC – if REPOX is used, this step is processed by the standalone server.
- Import the records into the OAI-Server.
- Automate the process as much as possible, ideally it should be fully automatic.

The advantages of this approach are: it is very easy to deploy, may take just a few hours, because the only effort required is to configure the software; it requires only free open source software and no in-house developers are needed, and consequently the least expensive option if it is appropriate. On the other side, this is the highest maintenance scenario in those cases where the metadata transfer can't be fully automated and may not be flexible enough to fit the library if the current systems that manage the data are too complex.

3.2.3.2 Tools

There are several tools available for this scenario which provide good options for the implementation of OAI-PMH:

- Rapid Visual OAI Tool can be used to graphically construct a OAI-PMH repository from a collection of files [12].

- The OAI-PMH installer[19] available in The European Library Handbook[18] - provided with crosswalks from UNIMARC and MARC21 to TEL-AP.
- REPOX[20] - An XML metadata repository to manage metadata sets with a standalone OAI-PMH server and a OAI-PMH client. It is designed to be a central piece of a Digital Library environment as all the records, including different versions of the same record, can be stored and managed there. It supports several metadata formats used by libraries and has a user interface to create metadata crosswalks. It is described in more detail in the next section.

3.2.3.3 The TELplus standalone OAI-PMH server

In work package 2 of TELplus, a standalone OAI-PMH server is being developed, and will be made available to all The European Library partners. It is an open source solution based on REPOX which aims to provide libraries with a simplified and flexible solution for exposing their data via OAI-PMH, without requiring specialized human resources in software development or system administration. Some of the characteristics that make it a suitable tool for The European Library partners are:

- Easy deployment – it will be provided with an easy installer that includes all other required software. It is also cross platform, since it is developed in Java it can be deployed in any operating system that has an available Java virtual machine.
- Support for several formats – initially it will support data import in ISO 2709 (including several variants), MarcXchange and MARCXML. Whenever possible, support will be added for other formats used by the libraries.
- Support for metadata crosswalks – it will have built in crosswalks for converting UNIMARC and MARC21 to TEL-AP and simple DC. Additionally, it will have an user interface to customize these crosswalks, and create new ones for other formats.

The first version is planed to be made available by the end of July 2008. It is foreseen that new versions will be made available during the course of the TELplus project when new requirements are identified, particularly the support for new data formats or character sets.

3.2.4 Other scenarios

In certain scenarios other kind of tools may be used. When building a repository that serves more purposes than just OAI, some full featured repository software may be used. The following examples already have OAI-PMH built-in:

- For setting up an e-print archive then it may be considered using the GNU EPrints software package.
- DSpace provides a digital asset management framework that includes preservation considerations.

3.3 Metadata crosswalks

All three scenarios require metadata to be converted to two other formats to be used in The European Library: The European Library application profile for objects (TEL-AP); and simple Dublin Core (DC). These are the minimum requirements for The European Library so it is possible to share data with other service providers

3.4 Testing

After the implementation is ready to run in order to test it, some OAI-PMH requests should be issued and sent to the OAI interface, and the responses checked.

Two tools may be used for testing purposes:

- the Repository Explorer at Vermont University [16], allows browsing and testing OAI-PMH repositories.
- The European Library OAI-PMH harvester[19] (available in The European Library Handbook) may be used to test the harvest of a server. This has the advantage of using exactly the same software that is used by The European Library portal.

3.5 Other implementation tasks

After the OAI-PMH implementation is completed and tested successfully, libraries may want to make their contents public and enable others to see their metadata. For that, the final step is to publish the OAI Server and sets in generic and/or specific OAI Data Provider groups. There are several web sites that can be used to register a Data Provider, like OAIster [13]. For more information on where to register Data Providers, see the Service Providers list in [14].

4 Conclusions

The purpose of this document is to clarify what is the OAI-PMH protocol and to give a detailed description and recommendation of the implementation possibilities, so that each library can use the most appropriate option for the Library Management System they currently have.

The OAI-PMH was driven by several factors, like the fast growth of the Internet, the slow traditional publishing model for the scholarly fields, a transfer of rights from author to publisher blocking the dissemination of results, the peer-review working as a barrier to new ideas as articles from prestigious institutions were favoured and subscription prices to publications becoming too high for libraries. The initial OAI target repositories were e-print archives, where author-submitted research papers are stored. The first prototype of OAI-PMH resulted from the Santa Fe Convention held to discuss and try to solve the problems described above.

OAI-PMH is a protocol oriented for interoperability between libraries. Two participant types are defined in OAI-PMH, the Data Providers which provide free access to metadata, and may offer free access to other resources, and Service Providers, entities who harvest data from Data Providers in order to provide higher-level services to users (e.g. searching, browsing, etc.). The success of OAI-PMH lies in its simplicity and effectiveness in sharing metadata with the harvesting model.

One of the goals of OAI-PMH is to provide an easy way to implement a low barrier solution for Data Providers. There are several characteristics which have brought it wide use. OAI-PMH is harvest oriented, hence its focus is on the transfer of metadata between participants, and common services like searching are outside its scope. Services that add value to the metadata may be implemented by Service Providers. There are two harvesting criteria: by datestamps (for creation, deletion or modification) and sets, collections of records defined by the provider. One of the greatest advantages of this protocol is its simplicity, the requests are simple HTTP GETs or POSTs and the responses are given in an XML format defined by the OAI. It supports multiple formats, but Dublin Core is mandatory for interoperability. It is scalable because there is no processing of metadata and access to the data is asynchronous.

There are only six actions (called verbs) that are used to access all that is needed from a Data Provider, which range from identifying a repository to listing records associated with a set. OAI-PMH has an approach that implicitly states that there are no real time requirements, since data accessed is not the most recent, because an intermediary provides the services, and not the data holders. This is not an issue in the scope of Digital Libraries because there aren't real time requirements for searching or browsing the metadata.

Three types of implementations were analysed for libraries to become Data Providers:

- a Library Management System module provided by a vendor, outsourcing the solution;
- an In-house OAI-PMH Server development, using existing tools to create an infrastructure to answer requests and access the data and develop on top of that;
- a Standalone OAI Server which should only need a configuration to connect to the sets and records to provide a complete solution.

The best option for each library partner in TELplus must take into account the local technological scenario and strategic option.

5 References

[1] The Santa Fe Convention for the Open Archives Initiative <http://www.openarchives.org/sfc/sfc_entry.htm>
[2] Open Archives Initiative Release Version 2.0 of the Protocol for Metadata Harvesting <http://www.openarchives.org/news/oaiv2press020614.html>
[3] The Open Archives Initiative Protocol for Metadata Harvesting v2.0 <http://www.openarchives.org/OAI/openarchivesprotocol.html>
[4] Dublin Core Metadata Initiative(DCMI) <http://dublincore.org/>
[5] Implementing OAI-PMH <http://www.oaforum.org/tutorial/english/page4.htm>
[6] OAICat Open Source Software (OSS) <http://www.oclc.org/research/software/oai/cat.htm>
[7] ibiblio's PHP, OAI-PMH Data Provider Implementation <http://www.ibiblio.org/oaibiblio/>
[8] oai-perl library <http://oai-perl.sourceforge.net/>
[9] ruby-oai <http://oai.rubyforge.org/>
[10] oaipmh Python package (pyoai) <http://www.infrae.com/download/oaipmh>
[11] OAI PMH Tools <http://www.openarchives.org/pmh/tools/tools.php>
[12] Rapid Visual OAI Tool <http://rvot.sourceforge.net/>
[13] OAIster <http://www.oaister.org/>
[14] Registered Service Providers <http://www.openarchives.org/service/listproviders.html>
[15] SRU/SRW <http://www.loc.gov/standards/sru/>
[16] Open Archives Initiative - Repository Explorer <http://re.cs.uct.ac.za/>
[17] Dienst Protocol Specification <http://www.cs.cornell.edu/cdlrg/dienst/protocols/DienstProtocol.htm>
[18] The European Library Handbook < http://www.theeuropeanlibrary.org/handbook/handbook.php?handbook_menu=0-5-1&handbook_menu_pg=handbook/Preparing_my_metadata/tel_application_profile.html >
[19] The European Library OAI-PMH installer & harvester <http://www.theeuropeanlibrary.org/handbook/handbook.php?handbook_menu=0-4-3-1&handbook_menu_pg=handbook/Accessing_my_collections/OAI-PMH_installer_harvester.html>
[20] Metadata Spaces: The Concept and a Case with REPOX <http://www.springerlink.com/content/qm51n87uw12877q6/>