



Harvest Service Expression: *harvest resource—oai-pmh basic*

1 Introduction

This service expression is a specialization of the harvest service genre [link to service genre]. It uses the OAI-PMH specification for harvesting metadata. All of the OAI-PMH operators (verbs) are supported. The resource (provider repository) being harvested exposes an interface for harvesting. Communications to the provider service implementation interface are represented in requests defined using HTTP. Requests and (dissemination) results are communicated between the client/requestor and the provider service implementation using HTTP, with results encoded in XML. The service expression supports disseminating unqualified Dublin Core v.1.1 metadata from the resource. The service implementation does not include a mechanism to authorize clients. How to discover service implementations that support harvesting is out of scope.

The words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in [RFC 2119].

The service expression description that follows uses e-Framework service expression description elements as of 2006-12-09. Other terms, e.g., client, provider, resource, are used as defined in the e-Framework. These definitions may conflict with those used in the OAI-PMH specification.

Items are tagged and identified using names assigned by the FRED project. Formal e-Framework names will be assigned by the e-Framework.

2 Service Expression Definition

2.1 Name

- FRED Service Expression Name: harvest resource—oai-pmh basic
- FRED ID: hdl:FREDNA/2EBE8F9695864E6C8C87C3A352526EAD
- e-Framework Service Expression Registry Name: TBD

2.2 Classification

Development Status: Proposed Developmental
 Prototype Production

Maturity: Immature Mature

Copyright © University of Southern Queensland and University of Memphis.



This work is licensed under the Creative Commons Attribution-Share Alike 2.5 Australia License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.5/au/>

This work is created as part of the Federated Repositories for Education (FRED) Project within the Australian ADL Partnership Laboratory. The FRED project is sponsored by the Australian Commonwealth Department of Education, Science and Training under the Framework for Open Learning Programme. The Australian ADL Partnership Laboratory is supported by the University of Southern Queensland.

The template structure and format of this document are based on e-Framework documentation templates and guidelines, which are governed by the e-Framework Intellectual Property Rights Statement <http://www.e-framework.org/Default.aspx?tabid=738>

Domain(s): Learning & Teaching Research
 Administration IT Services
 Libraries Common

Deployment Scale: Isolated Widespread

Purpose(s): Exemplar Application
 Modelling Toolkit

State Behaviour: Stateful Stateless

Transactional Behaviour: Transactional/ACID Non Transactional
 Transactional/Non-ACID

Batch Behaviour(s): Individual Batch

Time-Constraint Behaviour: Hard, Real Soft, Real None

Protocol Binding(s): Web Service SOAP REST
 HTTP Other

Service Genre Coverage: Full Extended
 Subset Overlapping

Service End Point: Provider Requestor
 Transcoder (provider & requestor)

Authentication/Authorization Dependency: Auth-Dependent
 Auth-Independent

2.3 Service Genre

- FRED Service Genre Name: harvest [[link to service genre](#)]
- e-Framework Service Genre Registry Name: TBD [[link to e-Framework service genre](#)]

2.4 Version

- FRED Version: 0.1 [[hdl:FREDNA/2EBE8F9695864E6C8C87C3A352526EAD](https://hdl.handle.net/2104/FREDNA/2EBE8F9695864E6C8C87C3A352526EAD)]
- e-Framework Service Expression Version: TBD

Version History

Version	Date	Author	Description
0.1	2006-12-10	DR	Initial version hdl:FREDNA/2EBE8F9695864E6C8C87C3A352526EAD
0.11	2006-12-19	NN	Taken out flags by Dan for work to be done in LOM WS version
0.12	2007-01-19	NN	Editorial; inserted full examples, attempted classification
0.13	2007-03-02	NN	Editorial: aligned this expression to fuller LOM WS expression

2.5 Description

The harvest resource—oai-pmh basic service expression specializes the harvest service genre through binding to the OAI-PMH specification. It further constrains the service implementations with additional constraints that are specific to the FRED community.

Provider repositories expose a "harvest" interface, defined by this service expression. Clients may send requests to this interface to gather the metadata stored in the repository. The OAI-PMH specification details some of the requirements on such an interface. This service expression follows OAI-PMH. Communications between the clients that harvest metadata from the repository and service implementation interface (repository interface) are through HTTP requests, not web services. Repositories are only required to expose Dublin Core metadata for objects in the repository. This service expression places additional requirements specific to the FRED community on all service implementations with the purpose of increasing interoperability.

Repository harvest interfaces are not access controlled, i.e., any client may attempt to harvest the metadata. There are no authentication controls. The provider is responsible for determining what results it will return. How a repository and service implementation decide how to respond to requests (i.e., what they expose) is not specified. The service expression only details the format and encoding of requests and responses, and specifics constraints and requirements on service implementations.

2.6 Functionality

As specified by OAI-PMH, the harvest resource service expression supports six distinct functions.

- **Identify:** Get descriptive information about the repository (the resource behind the service implementation interface) and the capabilities of the service implementation. This information enables a client to successfully communicate with the repository. Such information includes the repository's treatment of deleted records, and the granularity of its date stamps.
- **ListMetadataFormats:** Get the set of metadata formats that the repository is capable of providing as results for a harvest request. The repository may be capable of returning (disseminating) the metadata in different formats. This information enables a client to request metadata in a particular format.
- **ListSets:** Get information about the structure of the repository. Objects in the repository may be grouped into "sets". This information enables a client to selectively harvest data by specifying the parts of the repository to be harvested.
- **GetRecord:** Get the metadata for a single specific object in the repository. The client may request the desired metadata format. This function enables the client to get the metadata for a single item.
- **ListRecords:** Get metadata records from the repository. This function enables the client to get the metadata for a selected collection of items. The client may request selective harvesting by specifying a date range or by specifying a part of the repository (via "sets"). Metadata records are returned for the objects that match the selection criteria. Mechanisms exist to provide flow control so the results may be returned in chunks.
- **ListIdentifiers:** Get the headers of metadata records in the repository (record identifier, date stamp, sets the record belongs to, deletion status). This function enables the client to get summary information for a selected collection of items. The client may request selective results by specifying a date range or by specifying a part of the repository (via "sets"). Mechanisms exist to provide flow control so the results may be returned in chunks.

The service expression provides the capability to return unqualified Dublin Core metadata records for objects. A service implementation *MAY* support other metadata formats.

Communications to the service implementation is via requests encoded as HTTP methods and transmitted using HTTP.

No other functionality is provided. The functionality defined SHALL NOT be extended. The use of a qualified Dublin Core schema is an extension to the service expression which requires a new service expression to be written.

2.7 Usage Scenarios

Harvesting occurs to facilitate discovery of assets in the provider repository: discovery is transacted on the client repository instead, by means of the metadata harvested. The main reason to do this is when a single client repository gathers metadata from multiple provider repositories. In that case, discovery across the range of repositories involves only a single query on the client, rather than a separate query for each provider. The providers form a federation of repositories by having their metadata harvested and centralised. The federation can be ad hoc, with metadata harvested as found on open access repositories; or it can be formal, with policy guarantees such as uniform metadata profiles and service level agreements underpinning it. The typical scenario is:

1. Client *harvests* metadata instance for an item from the provider repository
2. Client *validates* the submitted metadata against the registry schema
3. Metadata is *associated* with content item

The *associate* step is necessary only if the identifier used at the client repository to refer to content is distinct from that used at the provider repository. If both repositories use the same global identifier, it is skipped.

Once a metadata record is discovered on the client through a query, the user may gain access to the content item referred to by the metadata record. This access occurs through a locator or identifier inside the metadata record harvested from the provider, possibly converted to the client's namespace through the *associate* step. The identifier for content is logically distinct from the identifier for the metadata record referring to it; the metadata record identifier is the parameter used by OAI-PMH. The content item need not reside on the provider repository.

The metadata on the client repository is expected to stay reasonably up to date. As a result, harvesting will usually be a periodically recurrent activity, and restricted only to those metadata records updated on the provider repository since the last harvesting session.

A client may not be interested in ingesting all the metadata records available from the provider, but only those relevant to the client repository subject matter. The provider offers a mechanism of restricting harvesting to those records identified as relevant; in OAI-PMH, this is done through "sets".

Harvesting proper is done through the ListRecords function. Of the other functions, Identify and ListMetadataFormats provide the client with the information necessary to set harvesting up, and choose the metadata profile relevant. (Since OAI-PMH requires unqualified Dublin Core to be supported by repositories for harvesting, the latter step is redundant for this service expression.) ListSets allows the client to restrict harvesting to the subsets of the provider repository contents it finds relevant. The scenario outlined fetches metadata as a batch rather than as individual records (GetRecord). It gathers complete metadata records, including local identifiers, in order to allow discovery on the client repository. The alternative is that it fetches only the headers of a set of records, e.g. recently updated records (ListIdentifiers), and either fetches the full records piecemeal, or makes the user browse them on the provider repository.

So, integrating these in on overall workflow:

A client acting on behalf of a federation of vocational education repositories harvests metadata from a participating provider.

- What version of OAI PMH does the provider support, and how does it handle date stamps and deleted records? (*Identify*)
- Given that: does the provider allow selective harvesting of only records describing vocational education modules? (*ListSets*; this presupposes an agreed vocabulary of sets, and that the provider may contain items other than vocational education)
- Alt 1: Request all headers of relevant metadata records. This is summary information about the metadata records, including when records have been updated and what the identifiers for the records are. The client may retrieve the corresponding records later, or redirect the end user to those metadata records on the provider. (*ListIdentifiers*)
- Alt 2: Request all relevant metadata records. These will be ingested into the client repository (*ListRecords*)
- Alt 3: Given the identifier for a metadata record (which may have been retrieved though *ListIdentifiers*), request the full corresponding metadata record. (*GetRecord*)

The service expression MAY be extended (by specifying the metadata format) if harvesting of other metadata formats is required.

The service expression is not applicable when the repository requires authentication to permit harvest. (NB: Such extensions require a new service expression.)

The service expression is not applicable when the repository filters the results that are returned based on authorization policies or rules that need to be communicated to the repository through the harvest interface. (NB: Such extensions require a new service expression.)

The service expression is not applicable if communications need to be secure. (NB: Such extensions require a new service expression.)

2.9 Requests & Behaviours

The format for request and responses are defined in the OAI-PMH specification. Six requests (verbs) SHALL be supported; these are the same requests enumerated under Functionality.

Identify (See OAI-PMH specification clause 4.2). The request and response SHALL be as specified. Additional requirements:

- The encoded repositoryName value (giving the human-readable name of the provider repository) SHALL NOT exceed 255 bytes in length.
- The protocolVersion value (version of OAI-PMH) SHALL be 2.0
- Any encoded adminEmail value (contact email of provider administrator) SHALL NOT exceed 255 bytes in length.
- The granularity value (minimum granularity of date stamp) SHALL be the finest granularity supported by the service implementation.

ListMetadataFormats (See OAI-PMH specification clause 4.4). The request and response SHALL be as specified. Additional requirements:

- The description of the metadataFormat SHALL be as follows:

```
<metadataFormat>
  <metadataPrefix>oai_dc</metadataPrefix>
  <schema>http://www.openarchives.org/OAI/2.0/oai_dc.xsd</schema>
  <metadataNamespace>
    http://www.openarchives.org/OAI/2.0/oai_dc</metadataNamespace>
</metadataFormat>
```

ListSets (See OAI-PMH specification clause 4.6). The request and response SHALL be as specified. Additional requirements:

- The encoded resumptionToken value SHALL NOT exceed 255 bytes in length.

GetRecord (See OAI-PMH specification clause 4.1). The request and response SHALL be as specified. Additional requirements:

- The encoded identifier value SHALL NOT exceed 255 bytes in length.

ListRecords (See OAI-PMH specification clause 4.5). The request and response SHALL be as specified. Additional requirements:

- The encoded resumptionToken value SHALL NOT exceed 255 bytes in length.

ListIdentifiers (See OAI-PMH specification clause 4.3). The request and response SHALL be as specified. Additional requirements:

- The encoded identifier value SHALL NOT exceed 255 bytes in length.
- The encoded resumptionToken value SHALL NOT exceed 255 bytes in length.

2.10 Use & Interactions

The model for a client to interact with a service implementation is defined in the OAI-PMH specification; see under Usage Scenarios for typical use of harvesting. Except as specified herein, the service implementation SHALL conform to the OAI-PMH specification.

The service implementation SHALL implement flow control (See OAI-PMH specification clause 3.5) with the following additional requirements:

- Flow control SHALL NOT be used if the number of records in the response is less than 1001. All records SHALL be returned in a single result set.
- The number of records in the results set SHALL NOT exceed 1000.
- The resumptionToken SHALL include an expirationDate. The specified expiration date of the resumptionToken SHALL be at least 10 minutes from the time the service implementation transmits the response to the client.

Responses for all requests SHALL include all applicable error codes (See OAI-PMH specification clause 3.6).

2.11 Structure

The data model for OAI-PMH harvests is given in the OAI-PMH specification §2. In brief:

- Repositories contain metadata describing content items (in OAI-PMH *resources*), which may or may not be stored in the same repositories.
- These metadata descriptions are stored or generated by **items** which reside on the repository.
- These metadata descriptions are stored or generated in the form of **records**.
- Repositories support one or more **metadata formats**.
- There may be several metadata records per item—i.e. each content item may be described by several metadata records. These records can differ from each other only in metadata format: an item can have at most one metadata record for each supported metadata format.
- Items are identified by unique **identifiers**.
- Therefore, the combination of identifier and metadata format identifies a unique metadata record.
- Items can be assigned to **sets**, which enable selective harvesting.
- Records are returned as XML; as specified in the OAI-PMH XML Schema, this contains:
 - A header, which in turn contains:
 - The item identifier
 - The **datestamp** for the record
 - The set membership of the item
 - An optional status attribute, indicating that the item has been deleted from the repository
 - The metadata record proper
 - An optional **about** container of data, which is metadata about the metadata—including e.g. a rights statement for the metadata record, or a description of the provenance of the metadata record

All OAI-PMH requests are read-only, and do not change the state of the provider repository. The List requests (ListRecords, ListIdentifiers, ListSets) allow for the possibility that the requested data may be too large to return in a single response. Instead of a single request query, OAI-PMH allows a List transaction to consist of multiple request–response pairs; each incomplete response will contain a resumption token, which the client must invoke to obtain the next segment of the response. The provider must therefore treat List requests as stateful: it generates a listing of sets or items in response to the query (and extracts identifiers or records from the items in turn), but it must remember how many

sets or items it has already presented to the client. Resumption tokens have optional expiration dates, so the provider MAY need to commit to holding state information on List requests for a significant period of time.

Between queries, the set of records that the initial request specified may change, as new records are created, modified or deleted; the provider MAY return an error indicating that the request should be reinitiated. If the request is honoured, the provider MUST return all unchanged records. It MAY update its listing of sets or items to reflect the intervening changes, excluding or including changed records.

Providers are expected to notify clients if any items have been deleted; this will allow the client to delete the corresponding record from its own repository. The provider does not necessarily notify the client when an item is deleted; rather, in the next cycle of harvesting, the records harvested by the client should include all identifiers for deleted items, along with the status indication that the items are now deleted. The client will use that list of items deleted since last harvest to delete its own instances of the same records. This means that the provider must retain a log of the item, even if the item itself has been deleted. This will allow it to generate a record in response to a harvest request for the item, indicating that the item is no longer available. Providers have the option of not retaining information on deleted items, retaining it temporarily, or retaining it permanently; they inform clients of how they treat deletion through the Identify request.

Providers are also expected to notify clients if any individual metadata records have been deleted (or become unavailable), even if the item generating them remains in the provider repository. If a provider discontinues LOM support, for instance, its LOM metadata records are deemed deleted, though its Dublin Core records for the same items remain in place. (As noted, a provider is required to maintain support for unqualified Dublin Core whatever other metadata format it chooses to support.)

Selective harvesting of records may be parameterised on date stamp, so that the client limits its request to those records created, modified or deleted within the given date span. A compliant provider MUST commit to keeping date stamps for all changes to the item resulting in an altered metadata record, and to exposing the mapping between date stamp and record (i.e. an index of items by most recent date stamp). This includes all creation and deletion of items, and all updates in metadata values, as well as all changes in metadata format (which will result in a different metadata record even if the content of the record is unchanged). Providers MAY have a date granularity of seconds rather than days.

Selective harvesting of records may be parameterized by set; this means that the provider must expose a mapping between set and record (i.e. an index of items by the set(s) they have been associated with).

2.12 Interface Definition

The format for OAI-PMH GET and POST HTTP methods are defined in the OAI-PMH specification. Both methods SHALL be supported. Other HTTP methods SHOULD NOT be supported.

The OAI-PMH response to a request SHALL be well-formed XML validating against the OAI-PMH XML Schema, as it is described in the OAI-PMH specification. The schema is available at: <http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd>. The schema allows a response to contain either data or an error report. Additional constraints on the sizes of XML elements in the result are defined in Behaviours & Requests elements within the service expression description. A base URL SHALL be specified as the content of the <request> element in the OAI-PMH response. The base URL SHALL be the URL of the target of the request.

The XML schema for unqualified Dublin Core SHALL follow the schema http://www.openarchives.org/OAI/2.0/oai_dc.xsd . The Dublin Core supported by this schema is version 1.1, as specified at <http://dublincore.org/documents/dces/>

Illustration:

GET Request

```
GET http://arrow.edu.au/oai?verb=GetRecord&identifier=hdl:1870/22FA672A787C4A0B82D8B0D7553ACE2B&metadataPrefix=oai_dc HTTP/1.1
```

POST Request

```
POST http://arrow.edu.au/oai? HTTP/1.1
Content-Length: 96
Content-Type: application/x-www-form-urlencoded
Date: Tue, 15 Nov 1994 08:12:31 GMT
Host: arrow.edu.au
User-Agent: Mozilla/5.0 (compatible; iCab 3.0.3; Macintosh; U; PPC Mac OS X)

verb=GetRecord&identifier=hdl:1870/22FA672A787C4A0B82D8B0D7553ACE2B&meta
taPrefix= oai_dc
```

Response

```
HTTP/1.1 200 OK
Content-Length: 96
Content-Type: application/text+html; charset=utf-8; action =
http://www.fred.net/wSDL/RegistryInterface#Identity
Accept: application/soap+xml
Date: Tue, 15 Nov 1994 08:12:35 GMT
Last-Modified: Tue, 15 Nov 1994 08:12:33 GMT

<?xml version="1.0" encoding="UTF-8"?>
<OAI:OAI-PMH xmlns:OAI="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>2002-05-01T19:20:30Z</responseDate>
  <request verb="GetRecord"
    identifier="hdl:1870/22FA672A787C4A0B82D8B0D7553ACE2B"
    metadataPrefix="oai_dc">arrow.edu.au/oai</request>
  <GetRecord>
    <record>
      <header>
        <identifier>hdl:1870/22FA672A787C4A0B82D8B0D7553ACE2B</identifier>
        <datestamp>1993-12-14</datestamp>
      </header>
      <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>Using Structural Metadata to Localize Experience of
            Digital Content</dc:title>
          <dc:creator>Dushay, Naomi</dc:creator>
          <dc:subject>Digital Libraries</dc:subject>
          <dc:description>With the increasing technical sophistication of
            both information consumers and providers, there is
            increasing demand for more meaningful experiences of digital
            information. We present a framework that separates digital
            object experience, or rendering, from digital object storage
            and manipulation, so the rendering can be tailored to
```

```

    particular communities of users.
  </dc:description>
  <dc:description>Comment: 23 pages including 2 appendices,
    8 figures</dc:description>
  <dc:date>2001-12-14</dc:date>
  </oai_dc:dc>
</metadata>
</record>
</GetRecord>
</OAI-PMH>

```

Except as specified herein, or in the OAI-PMH specification, the service implementation SHALL conform to the HTTP 1.1 specification.

- The GET and POST methods SHALL accept URIs of at least 4000 bytes in length. An HTTP status 414 Request-URI Too Long SHALL be returned if the length of the URI exceeds that which the service implementation can process.
- The POST method SHALL accept a message with Content-Length of at least 4000 bytes in length. An HTTP status 414 Request-URI Too Long SHALL be returned if the Content-Length exceeds that which the service implementation can process.
- Requests SHALL include a From request-header field with a well formed email address. The encoded email address SHALL NOT exceed 255 bytes. The service implementation SHALL return an HTTP status 400 Bad Request error if the field is missing or malformed.
- Requests SHALL include a User-Agent request-header field. The encoded User-Agent value SHALL NOT exceed 255 bytes. The service implementation SHALL return an HTTP status 400 Bad Request error if the field is missing.
- Responses SHALL include a no-cache pragma-directive.
- The service implementation MAY include load balancing. The service implementation may return an HTTP status 302 Found to redirect the client to a different service instance. The client SHOULD redirect the request to the indicated service instance.
- A service implementation SHALL return an HTTP status 503 Service Unavailable when it is too busy to respond to a request. The HTTP response SHALL include a Retry-After value. The client SHOULD obey the response and not retry the request until the specified time has expired. If the client retries the request too soon, the service implementation SHALL return an HTTP status 403 Forbidden.
- The service implementation SHALL support HTTP compression (See OAI-PMH specification clause 3.1.3). The service implementation SHALL support gzip compression.
- A service implementation SHALL return an HTTP status 400 BAD Request when the request contains code injection or other malicious elements.

2.13 Applicable Standards

The Open Archives Initiative Protocol for Metadata Harvesting, Protocol Version 2.0

<http://www.openarchives.org/OAI/openarchivesprotocol.html>

Dublin Core Metadata Element Set, Version 1.1: Reference Description

<http://dublincore.org/documents/dces/>

Hypertext Transfer Protocol -- HTTP/1.1

<http://www.ietf.org/rfc/rfc2616.txt>

2.14 Design Decisions & Tradeoffs

Consistency:

- The service implementation SHALL ensure that the time stamps for when content is added or modified are consistent with those returned to the client such that the metadata for all objects is harvestable.

Performance:

- A service implementation SHALL be capable of handling simultaneous requests from different clients.
- A service implementation SHOULD implement an indexing scheme or equivalent method to permit efficient harvesting by date ranges.
- Load balancing SHOULD be implemented for large repositories or those which are harvested frequently (continuously).
- The resumption token protocol of OAI-PMH means that a large result set must be harvested sequentially; so it is not parallelisable: a large result set may not be harvested by two harvesters in parallel, or out of sequence. The first harvesting of a large repository will therefore be time consuming, though it may not be time-intensive (resumption tokens may have a long time-to-live.)
- The harvest request to a set of repositories is trivially parallelisable: the same request to harvest may be made to several repositories at the same time. Ingesting several sets of metadata records returned simultaneously is a challenge for an Ingest service, but not a Harvest service.

Interoperability:

- Clients should be aware that URIs longer than 255 bytes may not be supported by intermediaries (caches, proxies).

Security:

- A service implementation SHALL inspect all requests for possible code injection.
- A client SHOULD validate all XML results against appropriate schemas.

2.15 Implementation Guide & Dependencies

Security Considerations:

- Service implementation may be subject to denial-of-service attacks.
- Administrator email addresses are returned to the client. Requests include a From request-header field with an email address. Care should be taken to maintain privacy of email addresses.
- Service implementation may log request and results. Security of logs should be maintained.
- There are no authorization or authentication controls. Care should be taken to maintain data privacy.
- While a service implementation may return provenance data, clients are not obliged to process it.
- Service implementations must respect flow control responses from repositories, including HTTP error codes such as 503 Service Unavailable and 302 Found.

Additional implementation guidance is available in:

Implementation Guidelines for the Open Archives Initiative Protocol for Metadata Harvesting

<http://www.openarchives.org/OAI/2.0/guidelines.htm>

with specific:

- *Guidelines for Repository Implementers*
<http://www.openarchives.org/OAI/2.0/guidelines-repository.htm>

- *Guidelines for Harvester Implementers*
<http://www.openarchives.org/OAI/2.0/guidelines-harvester.htm>
- *Guidelines for Aggregators, Caches and Proxies*
<http://www.openarchives.org/OAI/2.0/guidelines.htm>

2.16 Known Uses

Actual: None

Potential: The service expression could be used in a service usage model for federated metadata repositories. A federated metadata registry would be the client that would send harvest requests to the service implementations that provide harvest interfaces to the repositories in the federation. The requests would be used to gather the metadata used to populate the federation registry. The client would periodically harvest the repositories in the federation to obtain updates to the objects held in the repositories. The federated metadata registry could also provide a service implementation interface to allow other clients to harvest the metadata in the federation registry.

2.17 Service Expression Dependencies

None.

2.18 Related Service Expressions

- FRED Service Expression Name: harvest resource—oai-pmh lom ws, Vx.xx. The harvest resource—oai-pmh lom ws service expression provides the same core functionality as the harvest resource—oai-pmh basic service expression. The harvest resource—oai-pmh lom ws service expression is required to support LOM metadata, and communications are via web services, using SOAP over HTTP transport.

2.19 Related Service Usage Models

- FRED Service Usage Model: registry federation, Vx.xx. [link to service usage model] Harvest (genre) is a part of the registry federation service usage model (genre based) and is used to gather metadata from the repositories and collections that participate in the federation to build the registry data used for discovery.

2.20 Related Service Patterns

None.

