Advanced Overview of Version 2.0 of the Open Archives Initiative Protocol for Metadata Harvesting

Michael L. Nelson
Old Dominion University
Norfolk VA
mln@cs.odu.edu

Herbert Van de Sompel
Los Alamos National Laboratory
Los Alamos NM
herbertv@lanl.gov

Simeon Warner
Cornell University
Ithaca NY
simeon@cs.cornell.edu

ACM/IEEE Joint Conference on Digital Libraries
Tutorial 5
Portland, Oregon
14:00 - 17:30
July 14 2002

Scope and Focus

- This Tutorial is not…
  - an introduction to OAI-PMH
  - a listing of all the wonderful projects that use OAI-PMH
  - a discussion of the merits of metadata harvesting vs. distributed searching
- A passing familiarity is assumed for:
  - OAI-PMH 1.x
  - Dublin Core

Outline

- How 2.0 evolved from SFC and 1.x
  - people, processes, events
- What's new in 2.0
  - comparison with 1.x
- Guidelines, recommendations, best practices for 2.0 implementations
  - harvesters, repositories, aggregators, optional containers
- Novel applications of OAI-PMH

Spoiler: What's New in 2.0?!

- Good news: OAI-PMH is still

Six Verbs + DC

- Incremental improvements
  - single XML schema
  - ambiguities removed
  - more expressive options
  - cleaner separation of roles & responsibilities
- Bad news: not backwards compatible with 1.1

from
Santa Fe Convention
to
OAI-PMH v.2.0

	Santa Fe convention	OAI-PMH v.1.0/1.1	OAI-PMH v.2.0
nature	experimental	experimental	stable
verbs	Dienst	OAI-PMH	OAI-PMH
requests	HTTP GET/POST	HTTP GET/POST	HTTP GET/POST
responses	XML	XML	XML
transport	HTTP	HTTP	HTTP
metadata	OAMS	unqualified Dublin Core	unqualified Dublin Core
about	eprints	document like objects	resources
model	metadata harvesting	metadata harvesting	metadata harvesting

Santa Fe Convention [02/2000]

- goal: optimize discovery of e-prints
- input:
  - the UPS prototype
  - RePEc /SODA "data provider / service provider model"
  - Dienst protocol
  - deliberations at Santa Fe meeting [10/99]

OAI-PMH v.1.0 [01/2001]

- goal: optimize discovery of document-like objects
- input:
  - SFC
  - DLF meetings on metadata harvesting
  - deliberations at Cornell meeting [09/00]
  - alpha test group of OAI-PMH v.1.0

OAI-PMH v.1.0 [01/2001]

- low-barrier interoperability specification
- metadata harvesting model: data provider / service provider
- focus on document-like objects
- autonomous protocol
- HTTP based
- XML responses
- unqualified Dublin Core
- experimental: 12-18 months

Selected Pre- 2.0 OAI Highlights

- October 21-22, 1999 - initial UPS meeting
- February 15, 2000 - Santa Fe Convention published in D-Lib Magazine
  - precursor to the OAI metadata harvesting protocol
- June 3, 2000 - workshop at ACM DL 2000 (Texas)
- August 25, 2000 - OAI steering committee formed, DLF/CNI support
- September 7-8, 2000 - technical meeting at Cornell University
  - defined the core of the current OAI metadata harvesting protocol
- September 21, 2000 - workshop at ECDL 2000 (Portugal)
- November 1, 2000 - Alpha test group announced (~15 organizations)
- January 23, 2001 - OAI protocol 1.0 announced, OAI Open Day in the U.S. (Washington DC)
  - purpose: freeze protocol for 12-16 months, generate critical mass
- February 26, 2001 - OAI Open Day in Europe (Berlin)
- July 3, 2001 - OAI protocol 1.1 announced
  - to reflect changes in the W3C's XML latest schema recommendation
- September 8, 2001 - workshop at ECDL 2001 (Darmstadt)

OAI-PMH v.2.0 [06/2002]

- goal: recurrent exchange of metadata about resources between systems
- input:
  - OAI-PMH v.1.0
  - feedback on OAI-implementers
  - deliberations by OAI-tech [09/01 - 06/02]
  - alpha test group of OAI-PMH v.2.0 [03/02 - 06/02]
  - officially released June 14, 2002

OAI-PMH v.2.0 [06/2002]

- low-barrier interoperability specification
- metadata harvesting model: data provider / service provider
- metadata about resources
- autonomous protocol
- HTTP based
- XML responses
- unqualified Dublin Core
- stable

releasing OAI-PMH v.2.0

- creation of OAI-tech
- pre-alpha phase
- alpha-phase
- beta-phase

creation of OAI-tech [06/01]

- created for 1 year period
- charge:
    - review functionality and nature of OAI-PMH v.1.0
    - investigate extensions
    - release stable version of OAI-PMH by 05/02
    - determine need for infrastructure to support broad adoption of the protocol
- communication: listserv, SourceForge, conference calls

OAI-tech

US representatives

Thomas Krichel (Long Island U) - Jeff Young (OCLC) - Tim Cole - (U of Illinois at Urbana Champaign) - Hussein Suleman (Virginia Tech) - Simeon Warner (Cornell U) - Michael Nelson (NASA) - Caroline Arms (LoC) - Mohammad Zubair (Old Dominion U) - Steven Bird (U Penn.)

European representatives

Andy Powell (Bath U. & UKOLN) - Mogens Sandfaer (DTV) - Thomas Baron (CERN) - Les Carr (U of Southampton)

pre-alpha phase [09/01 – 02/02]

- review process by OAI-tech:
  - identification of issues
  - conference call to filter/combine issues
  - white paper per issue
  - on-line discussion per white paper
  - proposal for resolution of issue by OAI-exec
  - discussion of proposal & closure of issue
  - conference call to resolve open issues

pre-alpha phase [02/02]

- creation of revised protocol document
    - in-person meeting Lagoze - Van de Sompel - Nelson – Warner
    - autonomous decisions
    - *internal* vetting of protocol document

alpha phase [02/02 – 05/02]

- alpha-1 release to OAI-tech March 1st 2002
  - OAI-tech extended with alpha testers
  - discussions/implementations by OAI-tech
  - ongoing revision of protocol document

OAI-PMH 2.0 alpha testers (1/2)

- The British Library
- Cornell U. -- NSDL project & e-print arXiv
- Ex Libris
- FS Consulting Inc -- harvester for my.OAI
- Humboldt-Universität zu Berlin
- InQuirion Pty Ltd, RMIT University
- Library of Congress
- NASA
- OCLC

OAI-PMH 2.0 alpha testers (2/2)

- Old Dominion U. -- ARC , DP9
- U. of Illinois at Urbana-Champaign
- U. Of Southampton -- OAIA (now Celestial), CiteBase, eprints.org
- UCLA, John Hopkins U., Indiana U., NYU -- sheet music collection
- UKOLN, U. of Bath -- RDN
- Virginia Tech -- repository explorer

beta phase [05/02-06/02]

- beta release on May 1st 2002 to:
  - registered data providers and service providers
  - interested parties
- fine tuning of protocol document
- preparation for the release of 2.0 conformant tools by alpha testers

what's new in OAI-PMH v.2.0

- quick recap
- general changes to improve solidity of protocol
- corrections
- new functionality

Overview of OAI-PMH Verbs

	Verb	Function
metadata about the repository	Identify	description of archive
	ListMetadataFormats	metadata formats supported by archive
	ListSets	sets defined by archive
harvesting verbs	ListIdentifiers	OAI unique ids contained in archive
	ListRecords	listing of N records
	GetRecord	listing of a single record

most verbs take arguments: dates, sets, ids, metadata formats and resumption token (for flow control)

general changes

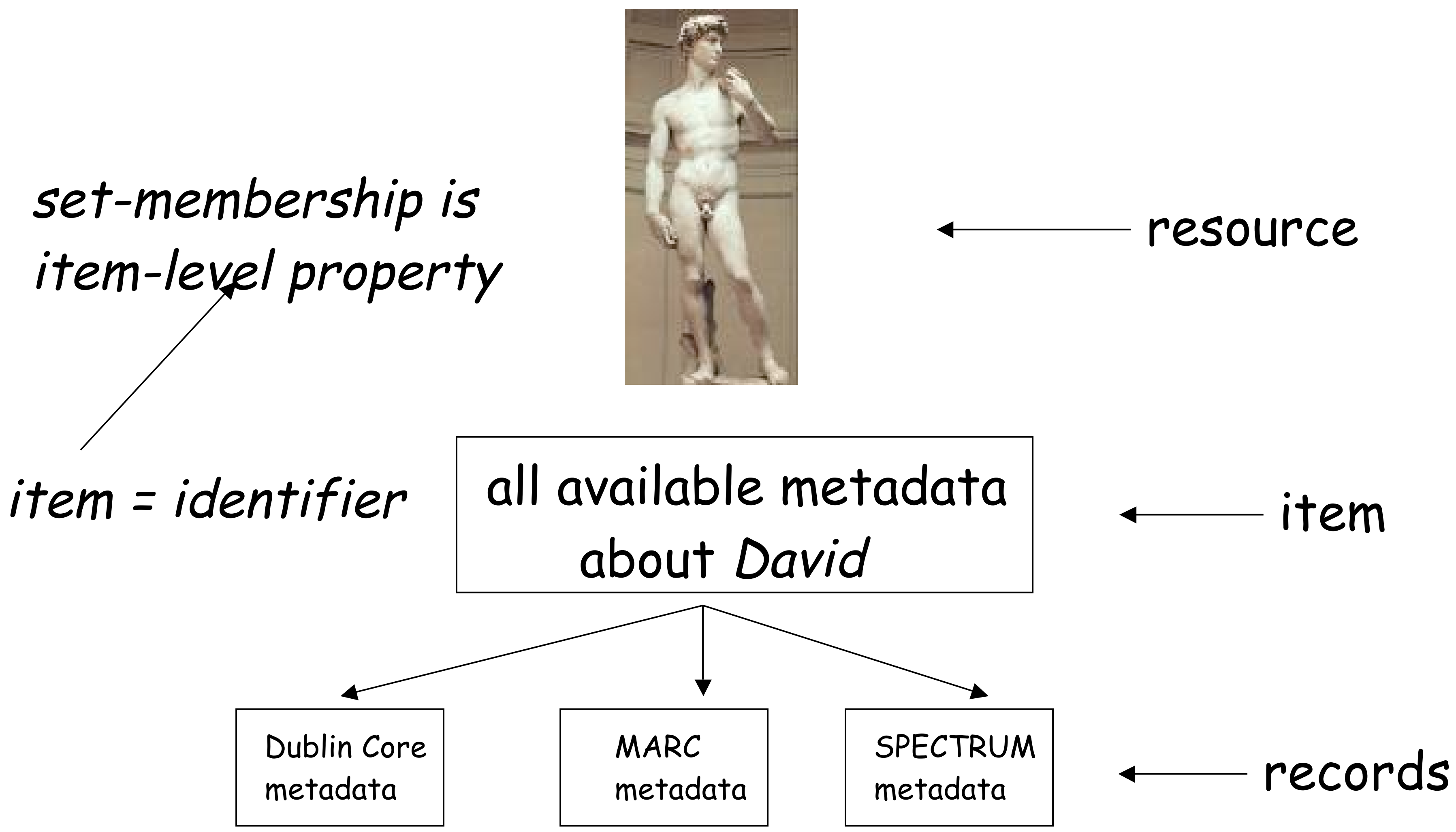
protocol vs periphery

- clear distinction between protocol and periphery
  - fixed protocol document
  - extensible implementation guidelines:
    - e.g. sample metadata formats, *description* containers, *about* containers
    - allows for OAI guidelines and community guidelines

OAI-PMH vs HTTP

- clear separation of OAI-PMH and HTTP
  - OAI-PMH error handling
    - all OK at HTTP level? => 200 OK
    - something wrong at OAI-PMH level? => OAI-PMH error (e.g. badVerb)
  - http codes 302, 503, etc. still available to implementers, but no longer represent OAI-PMH events

resource – item - record



record = identifier + metadata format + datestamp

other general changes

- better definitions of harvester, repository, item, unique identifier, record, set, selective harvesting
- oai_dc schema builds on DCMI XML Schema for unqualified Dublin Core
- usage of *must*, *must not* etc. as in RFC2119
- wording on response compression

other general changes

- all protocol responses can be validated with a single XML Schema
    - easier for data providers
    - no redundancy in type definitions
    - SOAP-ready
    - clean for error handling

response no errors

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH>
<responseDate>2002-0208T08:55:46Z</responseDate>
<request verb="GetRecord"… …>http://arXiv.org/oai2</request>
 <GetRecord>
  <record>
   <header>
    <identifier>oai:arXiv:cs/0112017</identifier>
    <datestamp>2001-12-14</datestamp>
    <setSpec>cs</setSpec>
    <setSpec>math</setSpec>
   </header>
   <metadata>
    …..
   </metadata>
  </record>
 </GetRecord>
</OAI-PMH>
```

note no http encoding of the OAI-PMH request

response with error

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH>
<responseDate>2002-0208T08:55:46Z</responseDate>
<request>http://arXiv.org/oai2</request>
<error code="badVerb">ShowMe is not a valid OAI-PMH verb</error>
</OAI-PMH>
```

with errors, only the correct attributes are echoed in <request>

corrections

dates/times

- all dates/times are UTC, encoded in ISO8601, Z-notation

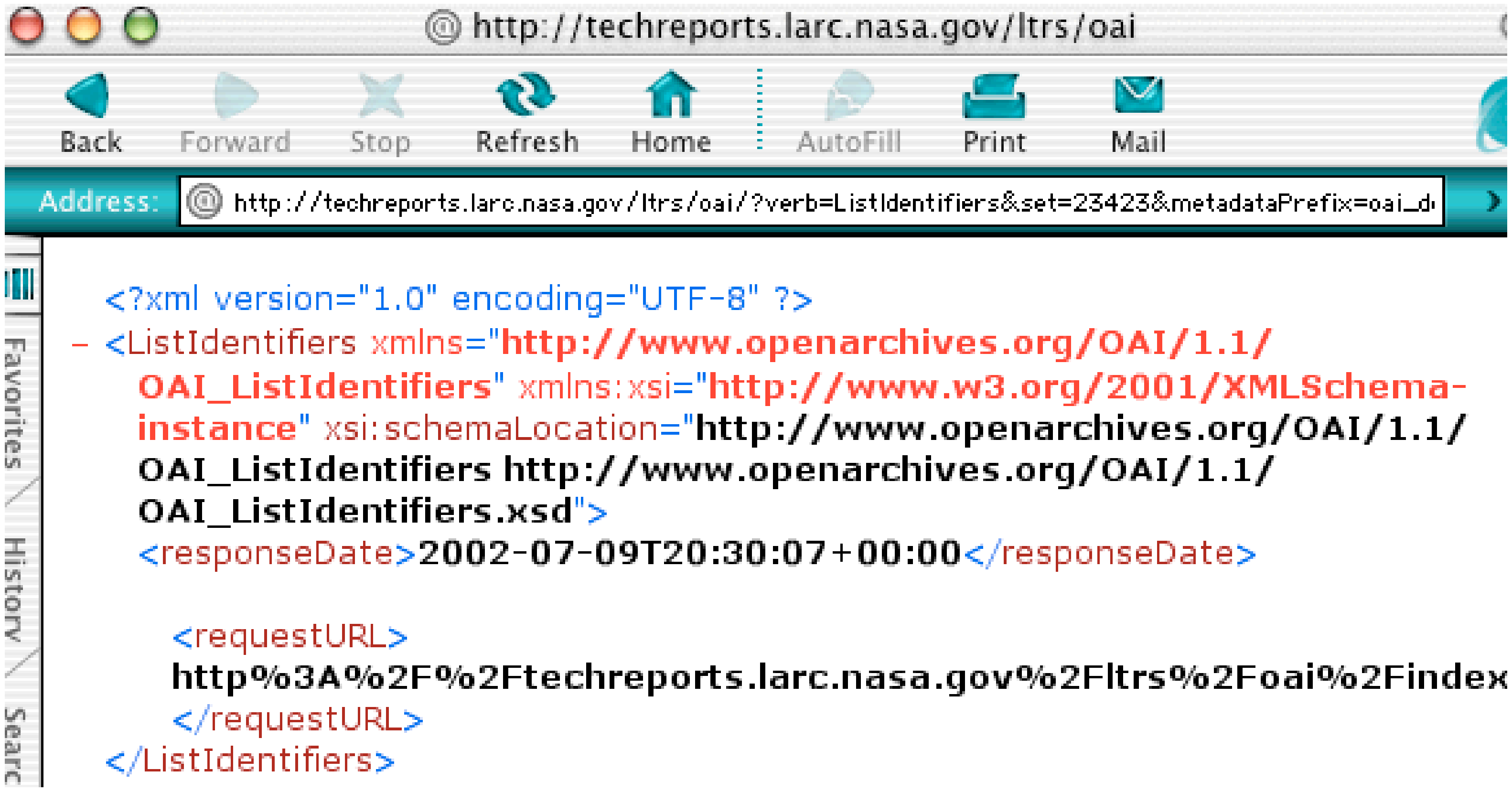
```
1957-03-20T20:30:00Z
```

resumptionToken

- idempotency of `resumptionToken`: return same incomplete list when `rT` is reissued
  - while no changes occur in the repo: strict
  - while changes occur in the repo: all items with unchanged `datestamp`
  - new, optional attributes for the resumptionToken:
    - `expirationDate`
    - `completeListSize`
    - `cursor`

noRecordsMatch

- 1.x - if no records match, an empty list was returned



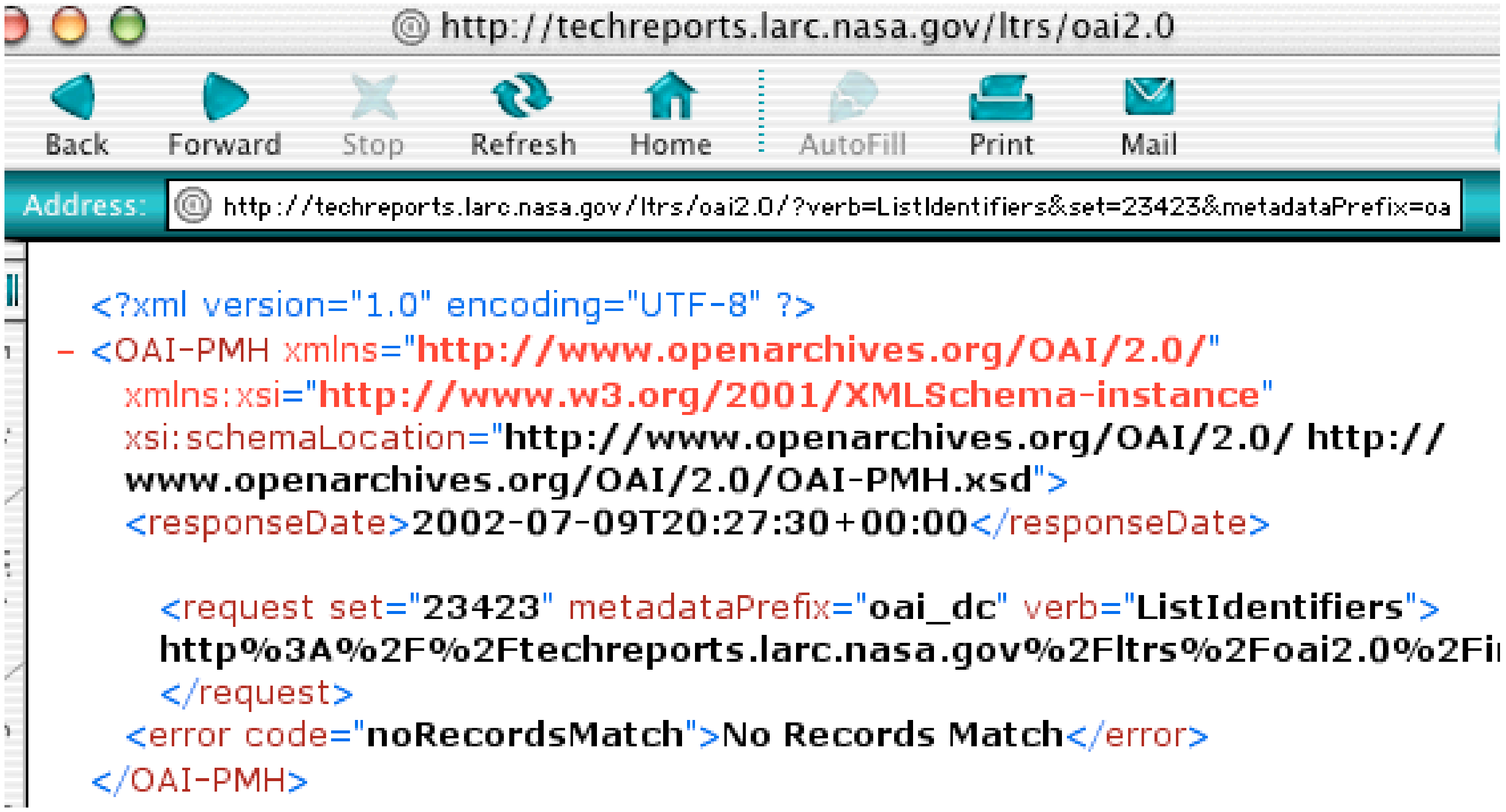
Address: http://techreports.larc.nasa.gov/ltrs/oai/?verb=ListIdentifiers&set=23423&metadataPrefix=oai_d

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <ListIdentifiers xmlns="http://www.openarchives.org/OAI/1.1/
   OAI_ListIdentifiers" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xsi:schemaLocation="http://www.openarchives.org/OAI/1.1/
   OAI_ListIdentifiers http://www.openarchives.org/OAI/1.1/
   OAI_ListIdentifiers.xsd">
   <responseDate>2002-07-09T20:30:07+00:00</responseDate>

     <requestURL>
     http%3A%2F%2Ftechreports.larc.nasa.gov%2Fltrs%2Foai%2Findex
     </requestURL>
  </ListIdentifiers>
```

noRecordsMatch

- 2.0 - if no records match, the error condition **noRecordsMatch** is returned -- not an empty list


http://techreports.larc.nasa.gov/ltrs/oai2.0
Back
Forward
Stop
Refresh
Home
AutoFill
Print
Mail
Address: http://techreports.larc.nasa.gov/ltrs/oai2.0/?verb=ListIdentifiers&set=23423&metadataPrefix=oa

```
  <?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>2002-07-09T20:27:30+00:00</responseDate>

      <request set="23423" metadataPrefix="oai_dc" verb="ListIdentifiers">
      http%3A%2F%2Ftechreports.larc.nasa.gov%2Fltrs%2Foai2.0%2Fi
      </request>
    <error code="noRecordsMatch">No Records Match</error>
  </OAI-PMH>
```

new functionality

harvesting granularity

- harvesting granularity
  - mandatory support of `YYYY-MM-DD`
  - optional support of `YYYY-MM-DDThh:mm:ssZ`
    - other granularities considered, but ultimately rejected
  - granularity of `from` and `until` must be the same

Identify

- `Identify` more expressive

```
<Identify>
    <repositoryName>Library of Congress 1</repositoryName>
    <baseURL>http://memory.loc.gov/cgi-bin/oai</baseURL>
    <protocolVersion>2.0</protocolVersion>
    <adminEmail>r.e.gillian@larc.nasa.gov</adminEmail>
    <adminEmail>rgillian@visi.net</adminEmail>
    <deletedRecord>transient</deletedRecord>
    <earliestDatestamp>1990-02-01T00:00:00Z</earliestDatestamp>
    <granularity>YYYY-MM-DDThh:mm:ssZ</granularity>
    <compression>deflate</compression>
```

header

- `header` contains set membership of item

```
<record>
  <header>
   <identifier>oai:arXiv:cs/0112017</identifier>
   <datestamp>2001-12-14</datestamp>
   <setSpec>cs</setSpec>
   <setSpec>math</setSpec>
  </header>
  <metadata>
   …..
  </metadata>
 </record>
```

eliminates the need for the “double harvest” 1.x required to get all records and all set information

ListIdentifiers

- `ListIdentifiers` **returns** `headers`

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH>
<responseDate>2002-0208T08:55:46Z</responseDate>
<request verb="…" …>http://arXiv.org/oai2</request>
<ListIdentifiers>
   <header>
    <identifier>oai:arXiv:hep-th/9801001</identifier>
    <datestamp>1999-02-23</datestamp>
    <setSpec>physic:hep</setSpec>
   </header>
   <header>
    <identifier>oai:arXiv:hep-th/9801002</identifier>
    <datestamp>1999-03-20</datestamp>
    <setSpec>physic:hep</setSpec>
    <setSpec>physic:exp</setSpec>
   </header>
    ……
```

ListIdentifiers

- `ListIdentifiers` **mandates** `metadataPrefix` **as argument**

```
http://www.perseus.tufts.edu/cgi-bin/pdataprov?
   verb=ListIdentifiers
   &metadataPrefix=olac
   &from=2001-01-01
   &until=2001-01-01
   &set=Perseus:collection:PersInfo
```

ListIdentifiers

- the changes to ListIdentifiers are subtle, and reflect a change in the OAI-PMH data model
- Could have been named “ListHeaders” or reduced to an option for ListRecords
  - “ListIdentifiers” kept for lexigraphical consistency

metadataPrefix

- character set for `metadataPrefix` and `setSpec` extended to URL-safe characters

```
A-Z a-z 0-9 _ ! ‘ $ ( ) + - . *
```

in the periphery

provenance

- introduction of `provenance` container to facilitate tracing of harvesting history

```
<about>
 <provenance>
  <originDescription>
   <baseURL>http://an.oa.org</baseURL>
   <identifier>oai:r1:plog/9801001</identifier>
   <datestamp>2001-08-13T13:00:02Z</datestamp>
   <metadataPrefix>oai_dc</metadataPrefix>
   <harvestDate>2001-08-15T12:01:30Z</harvestDate>
       <originDescription>
        … … …
       </originDescription>
  </originDescription>
 </provenance>
</about>
```

friends

- introduction of `friends` container to facilitate discovery of repositories

```
<description>
 <friends>
  <baseURL>http://cav2001.library.caltech.edu/perl/oai</baseURL>
  <baseURL>http://formations2.ulst.ac.uk/perl/oai</baseURL>
  <baseURL>http://cogprints.soton.ac.uk/perl/oai</baseURL>
  <baseURL>http://wave.ldc.upenn.edu/OLAC/dp/aps.php4</baseURL>
 </friends>
</description>
```

branding

- introduction of `branding` container for DPs to suggest rendering & association hints

```
<branding xmlns="http://www.openarchives.org/OAI/2.0/branding/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/branding/
                     http://www.openarchives.org/OAI/2.0/branding.xsd">
  <collectionIcon>
    <url>http://my.site/icon.png</url>
    <link>http://my.site/homepage.html</link>
    <title>MySite(tm)</title>
    <width>88</width>
    <height>31</height>
  </collectionIcon>
  <metadataRendering
    metadataNamespace="http://www.openarchives.org/OAI/2.0/oai_dc/"
    mimeType="text/xsl">http://some.where/DCrender.xsl</metadataRendering>
  <metadataRendering
    metadataNamespace="http://another.place/MARC"
    mimeType="text/css">http://another.place/MARCrender.css</metadataRendering>
</branding>
```

oai-identifier

- **revision of** `oai-identifier`

```
<description>
  <oai-identifier xmlns="http://www.openarchives.org/OAI/2.0/oai-
identifier"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai-
identifier
      http://www.openarchives.org/OAI/2.0/oai-identifier.xsd">
    <scheme>oai</scheme>
    <repositoryIdentifier>oai-stuff.foo.org</repositoryIdentifier>
    <delimiter>:</delimiter>
    <sampleIdentifier>oai:oai-stuff.foo.org:5324</sampleIdentifier>
  </oai-identifier>
</description>
```

domain based
repository names

oai_dc

- OAI 1.x: oai_dc Schema defined by OAI
- OAI 2.0: oai_dc Schema imports from DCMI Schema for unqualified DC elements

MARC21

- OAI 1.x: oai_marc
- OAI 2.0: **LoC marxml**, oai_marc
  - http://www.loc.gov/standards/marcxml/

did not make it into OAI-PMH v.2.0

- *SOAP implementation*
- Result set filtering
- Multiple / “best” metadata
- GetRecord -> GetRecords
- Machine readable rights management
- *XML format for “mini-archives”*

Detailed Review of the OAI-PMH 2.0 Verbs

Identify

1.1

- Arguments
  - none
- Errors
  - none

2.0

- Arguments
  - none
- Errors
  - badArgument

ListMetadataFormats

1.1

- Arguments
  - identifier (OPTIONAL)
- Errors
  - id does not exist

2.0

- Arguments
  - identifier (OPTIONAL)
- Errors
  - badArgument
  - noMetadataFormats
  - idDoesNotExist

ListSets

1.1

- Arguments
  - resumptionToken (EXCLUSIVE)
- Errors
  - no set hierarchy

2.0

- Arguments
  - resumptionToken (EXCLUSIVE)
- Errors
  - badArgument
  - badResumptionToken
  - noSetHierarchy

ListIdentifiers

1.1

- Arguments
  - from (OPTIONAL)
  - until (OPTIONAL)
  - set (OPTIONAL)
  - resumptionToken (EXCLUSIVE)
- Errors
  - no records match

2.0

- Arguments
  - from (OPTIONAL)
  - until (OPTIONAL)
  - set (OPTIONAL)
  - resumptionToken (EXCLUSIVE)
  - metadataPrefix (REQUIRED)
- Errors
  - badArgument
  - cannotDisseminateFormat
  - badResumptionToken
  - noSetHierarchy
  - noRecordsMatch

ListRecords

1.1

- Arguments
  - from (OPTIONAL)
  - until (OPTIONAL)
  - set (OPTIONAL)
  - resumptionToken (EXCLUSIVE)
  - metadataPrefix (REQUIRED)
- Errors
  - no records match
  - metadata format cannot be disseminated

2.0

- Arguments
  - from (OPTIONAL)
  - until (OPTIONAL)
  - set (OPTIONAL)
  - resumptionToken (EXCLUSIVE)
  - metadataPrefix (REQUIRED)
- Errors
  - noRecordsMatch
  - cannotDisseminateFormat
  - badResumptionToken
  - noSetHierarchy
  - badArgument

GetRecord

1.1

- Arguments
  - identifier (REQUIRED)
  - metadataPrefix (REQUIRED)
- Errors
  - id does not exist
  - metadata format cannot be disseminated

2.0

- Arguments
  - identifier (REQUIRED)
  - metadataPrefix (REQUIRED)
- Errors
  - badArgument
  - cannotDisseminateFormat
  - idDoesNotExist

Argument Summary

	metadataPrefix	from	until	set	resumptionToken	identifier
Identify	✗	✗	✗	✗	✗	✗
ListMetadata Formats	✗	✗	✗	✗	✗	optional
ListSets	✗	✗	✗	✗	exclusive	✗
ListIdentifiers	✓	optional	optional	optional	exclusive	✗
ListRecords	✓	optional	optional	optional	exclusive	✗
GetRecord	✓	✗	✗	✗	✗	✓

Error Summary

Identify	BA						
ListMetadata Formats	BA					NMF	IDDNE
ListSets	BA	BRT			NSH		
ListIdentifiers	BA	BRT	CDF	NRM	NSH		
ListRecords	BA	BRT	CDF	NRM	NSH		
GetRecord	BA		CDF				IDDNE

Generate badVerb on any input not matching the 6 defined verbs

this is an inversion of the table in section 3.6 of the OAI-PMH specification

Repository
Implemenation Guidelines

Minimal Repository

- 2.0 provides many expressive, but optional features
  - but still low barrier!
- if you are writing your own repository software, the quickest path to implementation can involve initially:
  - only supporting DC
  - skipping: <about>, sets, compression
  - skip flow control (resumptionTokens) if < 1000 items
- add optional features as requirements and familiarity allows

Be Honest with datestamp!

- a change in the process of dynamic generation of a metada format *really does mean all records have been updated!*

```
if (internalItemDatestamp > disseminationInterfaceDatestamp) {
  datestamp = internalItemDatestamp
} else {
  datestamp = disseminationInterfaceDatestamp
}
```

Not Hiding Updates

- OAI-PMH is designed to allow incremental harvesting
- Updates must be available by the end of the period of the datestamp assigned, i.e.
  - Day granularity => during same day
  - Seconds granularity => during same second
- Reason: harvesters need to overlap requests by just one datestamp interval (one day or one second)
  - in 1.x, 2 intervals were required (in many circumstances)

State in resumptionTokens

- HTTP is stateless
- resumptionTokens allow state information to be passed back to the repository to create a complete list from sequence of incomplete lists
- EITHER – all state in resumptionToken
- OR – cache result set in repository

Caching the Result Set

- Repository caches results of initial request, returns only incomplete list
- resumptionToken does not contain all state information, it includes:
  - a session id
  - offset information, necessary for idempotency
- resumptionToken allows repository to return next incomplete list
- increased complexity due to cache management
  - but a potential performance win

All State in the resumptionToken

- Arrange that remaining items/headers in complete list response can be specified with a new query and encode that in resumptionToken
- One simple approach is to return items/headers in id order and make the new query specify the same parameters and the last id return (or by date)
  - simple to implement, but possibly longer execution times
- Can encode parameters very simply:

```
<resumptionToken>metadataPrefix=oai_dc
  from=1999-02-03&until=2002-04-01&
  lastid=fghy:45:123</resumptionToken>
```

resumptionToken attributes (1)

- `expirationDate` – likely to be useful when cache clean-up schedule is known
  - Do not specify `expirationDate` if all state in `resumptionToken`
- `badResumptionToken` error to be used if `resumptionToken` expired
  - May also be used if request cannot be completed for some other reason
    - e.g.: if repository changes cause the incomplete list to have no records

resumptionToken attributes (2)

- `completeListSize` and `cursor` optionally provide information about size of complete list and number of records so far disseminated
  - use consistently if used
  - designed for status monitoring
  - caveat harvester: `completeListSize` may be approximate and may be revised

resumptionToken

The only defined use of resumptionToken is as follows:

- a repository **must** include a resumptionToken element as part of each response that includes an incomplete list;

- in order to retrieve the next portion of the complete list, the next request **must** use the value of that resumptionToken element as the value of the resumptionToken argument of the request;

- the response containing the incomplete list that completes the list **must** include an empty resumptionToken element;

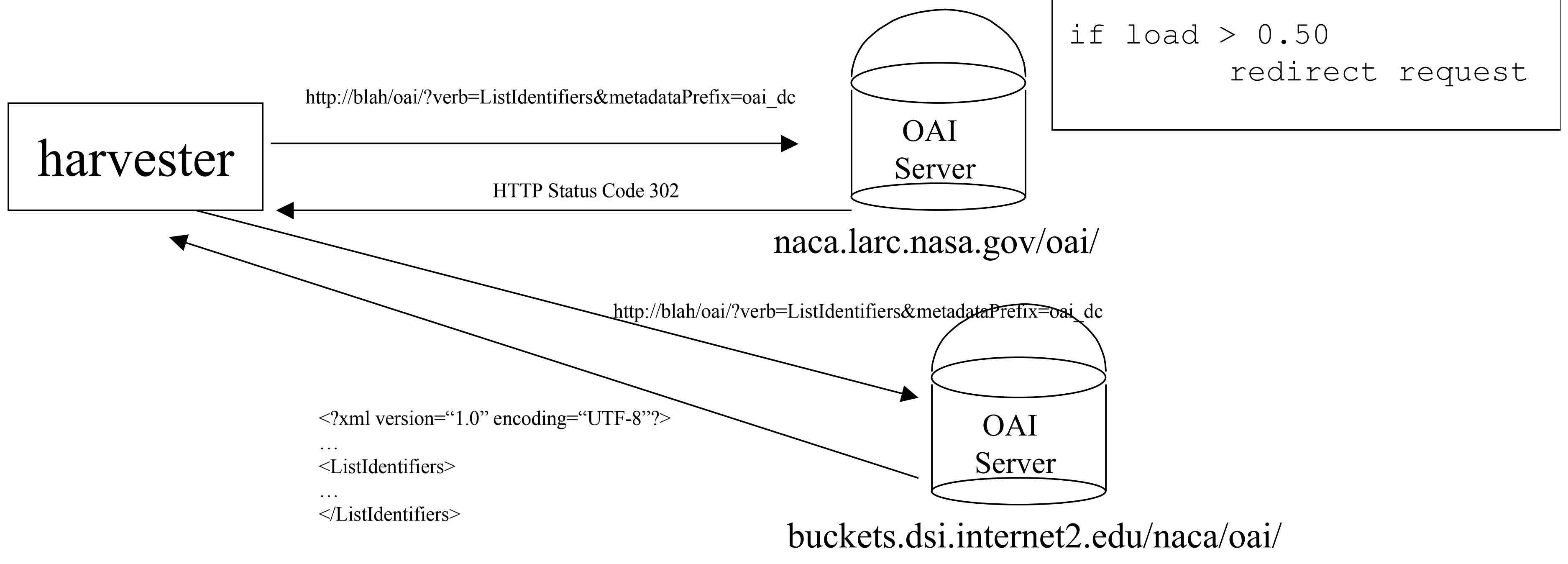
Flow Control & Load Balancing

How to respond to a “bad” harvester:

1. HTTP status code 200; response to OAI-PMH request with a resumptionToken.
2. HTTP status code 503; with the Retry-After header set to an appropriate value if subsequent request follows too quickly or if the server is heavily loaded.
3. HTTP status code 403; with an appropriate reason specified if subsequent requests do not adhere to Retry-After delays.

302 Load Balancing

- Interactive users on main DL machine should not be impacted by metadata harvesting
  - don't take deliveries through the front door
  - not part of the protocol; defined outside the protocol


harvester
http://blah/oai/?verb=ListIdentifiers&metadataPrefix=oai_dc
OAI Server
HTTP Status Code 302
naca.larc.nasa.gov/oai/
http://blah/oai/?verb=ListIdentifiers&metadataPrefix=oai_dc
OAI Server
buckets.dsi.internet2.edu/naca/oai/

```
if load > 0.50
        redirect request
```

```
<?xml version="1.0" encoding="UTF-8"?>
…
<ListIdentifiers>
…
</ListIdentifiers>
```

DNS Load Balancing

- using a DNS rotor, establish
  - a.foo.org, b.foo.org, c.foo.org
  - each with a synchronized copy of the repository
  - let DNS & chance distribute the load

Load Balancing Caveats

- Copies of the repository must be synchronized
  - (cf. Pande, et al. JCDL 02)
- Complex hierarchies are possible
  - programmer must insure no cycles in redirection graphs!
- The baseURL in the reply must always point to the original repository, not the repository that eventually answered the request

Error Handling: Verbosity

More is better…

```
<error code="badArgument">Illegal argument ‘foo’</error>
<error code="badArgument">Illegal argument ‘bar’</error>
```

is preferred over:

```
<error code="badArgument">Illegal arguments ‘foo’, ‘bar’</error>
```

which is preferred over:

```
<error code="badArgument">Illegal arguments</error>
```

Error Handling: Levels

- the OAI-PMH error / exception conditions are for OAI-PMH semantic events
- they are not for situations when:
  - the database is down
  - a record is malformed
    - remember: record = id + datestamp + metadataPrefix
    - if you're missing one of those, you don't have an OAI record!
  - and other conditions that occur outside the OAI scope
    - use http codes 500, 503 or other appropriate values to indicate non-OAI problems

Error Handling: Extensions

- Arguments that are not 'required', 'optional' or 'exclusive' are 'illegal' and should generate `badArgument` errors.
- If you want to extend the OAI-PMH:
  - stop and consider: do you really need to?
    - maybe you should have different OAI-PMH interfaces, or creative metadata formats
  - if you really, really want to, tunnel your extensions through the "set" feature
    - see http://www.dlib.org/dlib/december01/suleman/12suleman.html for examples

Idempotency of “List” Requests (1)

- Purpose is to allow harvesters to recover from lost responses or crashes without starting a large harvest from scratch
- Recover by re-issuing request using `resumptionToken` from previous request
- IMPLICATION: harvester must accept both the most recent `resumptionToken` issued and the previous one

Idempotency of "List" Requests (2)

- response to a re-issued request must contain all unchanged records
- any changed records will get new datestamps after time of initial request
- changes will be picked up by subsequent harvest if not included

[no experience yet with incomplete responses to ListSets or ListMetadataFormats requests]

Case Study: “bucket” based repositories

- Buckets: see Nelson & Maly, CACM 44(5)
- 2.0
  - LTRS - techreports.larc.nasa.gov/ltrs/oai2.0/ (file system, refer)
  - NACA - naca.larc.nasa.gov/oai2.0/ (file system, refer)
- 1.1
  - LTRS - techreports.larc.nasa.gov/ltrs/oai/
  - NACA - naca.larc.nasa.gov/ltrs/oai/
  - Open Video - www.open-video.org/oai/ (MySQL, local)
  - JTRS - ston.jsc.nasa.gov/collections/TRS/oai (MS Access dump, local)
  - GLTRS (filesystem, HTML scraping)
- Characteristics:
  - resumptionToken support initially skipped; added later (all)
    - highly encoded rT’s: [2001-01-01!!!!301!600]
  - sets initially skipped, added later (LTRS)
  - initially had load balancing with 2 NACA repositories…

Case Study: “bucket” based repositories

- in bucket terminology:
  - 6 OAI verbs (methods) added to the existing list of methods
    - http://naca.larc.nasa.gov/oai2.0/?method=list_methods
    - http://naca.larc.nasa.gov/oai2.0/?method=list_source&target=ListIdentifiers
  - a data element is added to the bucket that contains the specifics of the particular repository and its metadata format
    - http://naca.larc.nasa.gov/oai2.0/?method=display&pkg_name=oai&element_name=oai.pl

Harvester Implementation Guidelines

Be a Polite OAI Neighbor

- Re-use existing free harvesters rather than writing your own
  - `http://www.openarchives.org/tools/index.html`
- Read `http://www.robotstxt.org/wc/robots.html` if you insist on writing your own
- Provide meaningful `User-Agent` & `From` http headers
  - these values can be configured even if using someone else's harvester

Listen to the Repository!

- Check Identify's `<granularity>` element if you wish to use finer than YYYY-MM-DD
- If you harvest with sets, remember that ":" indicates hierarchies
  - harvesting "a" will recursively harvest "a:b", "a:b:c", and "a:d"
- Check for and handle non-200 http status codes (503, 302, etc.)
- empty resumptionToken => end of complete list
- consider asking for compressed responses if the repository supports them…

How to Grab *Everything*

- Issue an **Identify** request to find the finest datestamp granularity supported.
- Issue a **ListMetadataFormats** request to obtain a list of all metadataPrefixes supported.
- Harvest using **ListRecords** requests for each metadataPrefix supported. Knowledge of the datestamp granularity allows for less overlap if granularities finer than a day are supported.
- Set structure can be inferred from the setSpec elements in the header blocks of each record returned (consistency checks are possible).
- Items may be reconstructed from the constituent records. Local datestamps must be assigned to harvested items.
- Provenance and other information in `<about>` blocks may be re-assembled at the item level if it is the same for all metadata formats harvested. However, this information may be supplied differently for different metadata formats and may thus need to be store separately for each metadata format.

Harvesting v1.1 and v2.0

- Not difficult to handle both cases, test with `Identify`:
  - v1.1: `<Identify>` `<protocolVersion>`
  - v2.0 `<OAI-PMH>` `<Identify>` `<protocolVersion>`
- Note also different error and exception handling
- Many similarities, harvesters can share lots of code

Harvesting Demo

- Harvester written in Perl
- Handles v1.0, v1.1 and v2.0
- Uses `LWP`, `Expat` and `XML::Parser` (no schema validation)
- UTF-8 conditioning to deal with some “imperfect” repositories
- Sequence of requests: `Identify`, `ListMetadataFormats`, `ListSets` then `ListRecords`/`ListIdentifiers`
- Support for incremental harvesting, uses `responseDate` from last harvest to get new start `datestamp`

Harvesting logs

- Alan Kent's v2.0 harvester logs: http://www.inquirion.com:8123/public/collList;collListCmd=list
- Alan Kent's summary of v1.1 harvesting results http://www.mds.rmit.edu.au/~ajk/oai/interop/summary.htm
- Celestial v1.1 harvesting logs http://celestial.eprints.org/cgi-bin/status
- DP9 gateway using arc harvested information http://arc.cs.odu.edu:8080/dp9/index.jsp

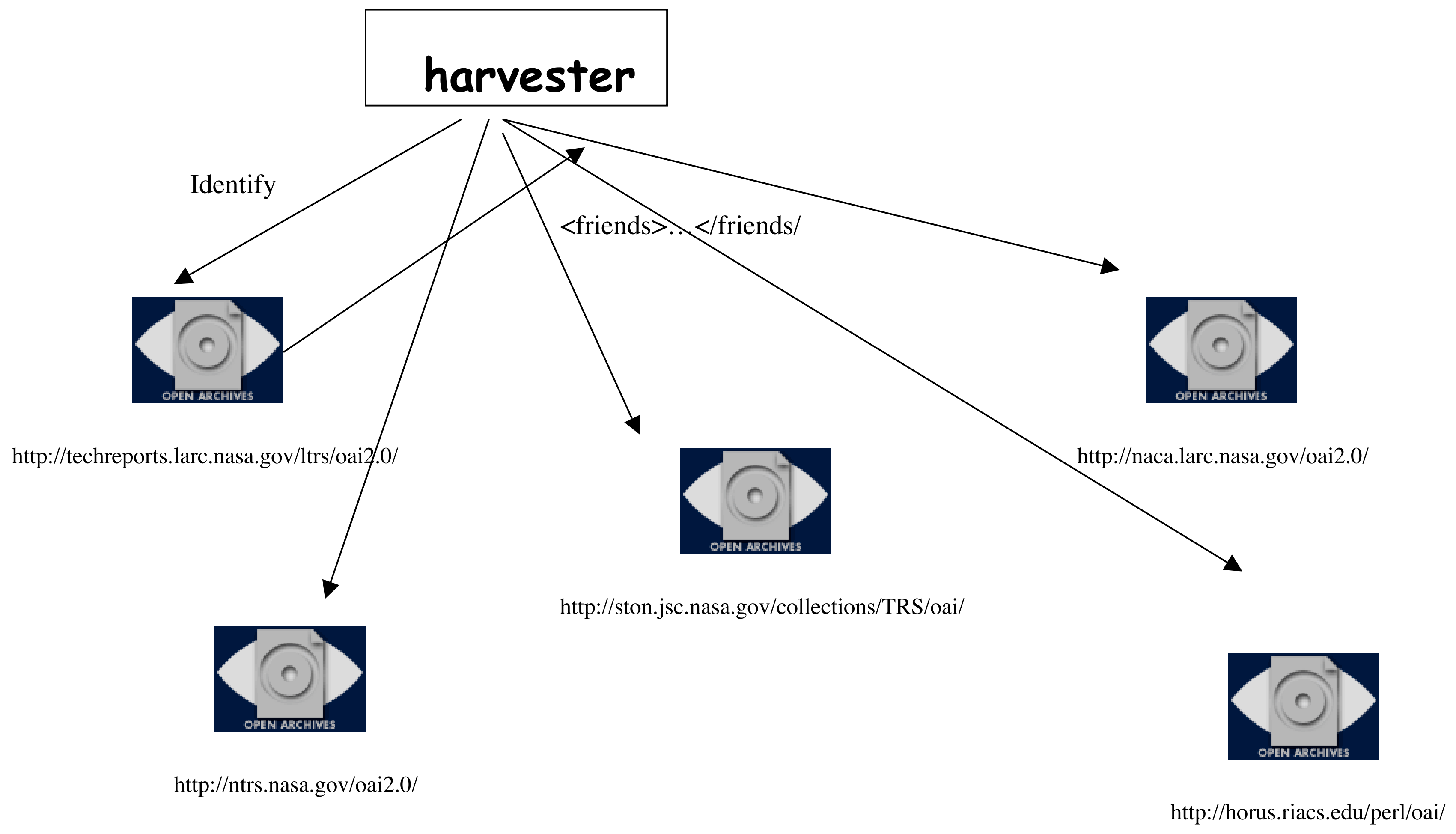
NASA `<friends>` example (1)

- A light weight, DP-centric method to communicate the existence of "others"

http://techreports.larc.nasa.gov/ltrs/oai2.0/?verb=Identify

```
..
<description>
 <friends ..namespace stuff..>
   <baseURL>http://naca.larc.nasa.gov/oai2.0</baseURL>
   <baseURL>http://ntrs.nasa.gov/oai2.0</baseURL>
   <baseURL>http://horus.riacs.edu/perl/oai/</baseURL>
   <baseURL>http://ston.jsc.nasa.gov/collections/TRS/oai/</baseURL>
 </friends>
</description>
..
```

NASA <friends> example (2)


harvester
Identify
<friends>...</friends/
OPEN ARCHIVES
http://techreports.larc.nasa.gov/ltrs/oai2.0/
http://ntrs.nasa.gov/oai2.0/
http://ston.jsc.nasa.gov/collections/TRS/oai/
http://naca.larc.nasa.gov/oai2.0/
http://horus.riacs.edu/perl/oai/

Case Study: arXiv (1)

- Existing system, running >10years. Written mostly in Perl
- Flat file system for 'database'
- 200k papers with metadata in homebrew format
- ~200 updates/day. OAI repository just one view of system, must integrate with daily update schedule

Case Study: arXiv (2)

- Write in Perl
  - Easy integration with rest of system, reuse code from v1.0/v1.1 interface
  - Use `libwww; XML::DOM`
- Daily rebuild of datestamp database
  - No existing date in system appropriate
  - Base on Unix `cdate` of metadata files
- On-the-fly metadata translation
  - Straightforward, avoids data duplication

Case Study: arXiv (3)

- Flow control to avoid loading server and to avoid harvesters tripping robot alarms
  - `resumptionTokens` to limit response size (500-1000 records or 5k-10k identifiers / response)
  - 503 `Retry-After` replies based on client ip
- Implement `resumptionTokens` that include all state
  - *Avoid need to cache result sets / clean cache*

Aggregator / Cache / Proxy Implementation Guidelines

`<provenance>` & datestamps

- reminder: datestamps are local to the repository, and an re-exporting service must use new local datestamps
- such services should use the `<provenance>` container to preserve the original datestamps

Identifiers are Local

- identifiers are local to the repository
- unless you *absolutely* did not change the metadata *and* the identifier corresponds to a recognized URI scheme, use a new identifier upon re-exporting
  - use the `<provenance>` container to preserve the harvesting history

Derived from the same item?

3 ways to determine if records share provenance from the same item:

1. both records have the same identifier *and* the baseURL in the request elements of the OAI-PMH reponses which include the record are the same;
2. both records have the same identifier *and* that identifier belongs to some recognized URI scheme;
3. the provenance containers of both records have the same entries for both the identifier and baseURL;

`<provenance>` example (1)

Consider a request from `crosswalker.oa.org`:

```
http://odd.oa.org?verb=GetRecord
  &identifier=oai:odd.oa.org:z1x2y3&metadataPrefix=odd_fmt
```

and the following response from `odd.oa.org`:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetRecord ..namespace stuff..>
  <responseDate>2002-02-08T08:55:46.1</responseDate>
  <request verb="GetRecord" metadataPrefix="odd_fmt"
     identifier="oai:odd.oa.org:z1x2y3">http://odd.oa.org</request>
  <record>
    <header>
      <identifier>oai:odd.oa.org:z1x2y3</identifier>
      <datestamp>1999-08-07T06:05:04Z</datestamp>
    </header>
    <metadata> ...metadata record in odd_fmt... </metadata>
  </record>
</GetRecord>
```

`<provenance>` example (2)

Imagine that `crosswalker.oa.org` cross-walks harvested metadata from `odd_fmt` into `oai_marc` and then re-exposes the metadata with new identifiers.

A request from `getmarc.oa.org`:

```
http://crosswalker.oa.org?verb=GetRecord
  &identifier=oai:cw.oa.org:z1x2y3
  &metadataPrefix=oai_marc
```

might then yield the following response from `crosswalker.oa.org`:

<provenance> example (3)

```
<record>
  <header> <identifier>oai:cw.oa.org:z1x2y3</identifier>
           <datestamp>2002-02-09T01:15:43Z</datestamp>
  </header>
  <metadata> ...metadata record in oai_marc... </metadata>
  <about>
    <provenance ..namespace stuff..>
     <originDescription harvestDate="2002-02-08T08:55:46Z"
                        altered="true">
        <baseURL>http://odd.oa.org</baseURL>
        <identifier>oai:odd.oa.org:z1x2y3</identifier>
        <datestamp>1999-08-07T06:05:04Z</datestamp>
        <metadataNamespace>http://odd.oa.org/odd_fmt</..>
      </originDescription>
    </provenance>
  </about>
</record>
```

`<provenance>` example (4)

This `oai_marc` record is then re-exposed by `getmarc.oa.org` with the same identifier `oai:cw.oa.og:z1x2y3` (because the record has not been altered).

The associated `<provenance>` container might be:

<provenance> example (5)

```
<record>
  <header> <identifier>oai:cw.oa.org:z1x2y3</identifier>
           <datestamp>2002-03-01T01:46:11Z</datestamp> </header>
  <metadata> ...metadata record in oai_marc... </metadata>
  <about>
    <provenance ..namespace stuff..>
      <originDescription harvestDate="2002-03-01T01:23:45"
                         altered="false">
        <baseURL>http://crosswalker.oa.org/<baseURL>
        <identifier>oai:cw.oa.org:z1x2y3</identifier>
        <datestamp>2002-02-09T01:15:43Z</datestamp>
        <metadataNamespace>http://../oai_marc</..>
          <originDescription harvestDate="2002-02-08T08:55:46Z" altered="true">
            <baseURL>http://odd.oa.org</baseURL>
            <identifier>oai:odd.oa.org:z1x2y3</identifier>
            <datestamp>1999-08-07T06:05:04Z</datestamp>
            <metadataNamespace>http://odd.oa.org/odd_fmt</..>
          </originDescription>
      </originDescription>
    </provenance>
  </about>
</record>
```

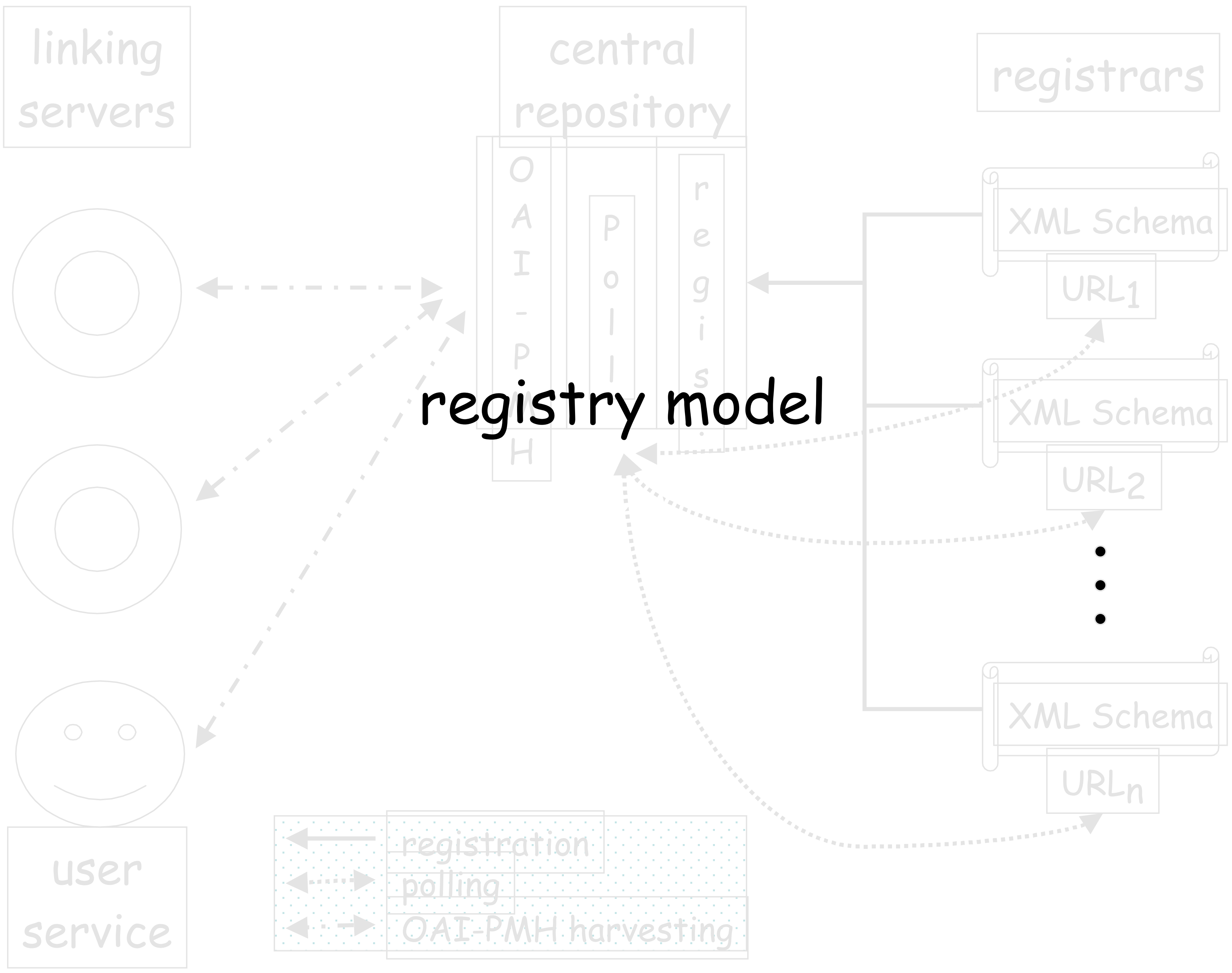
oai-identifier

- not compatible with v1.1 oai-identifier
  - repositoryName now domain name based
  - not reliant upon OAI centralized registration
- one-to-one mapping for escaping characters: %3F allowed, %3f not
  - allows simple comparison

looking ahead:
novel uses of OAI-PMH

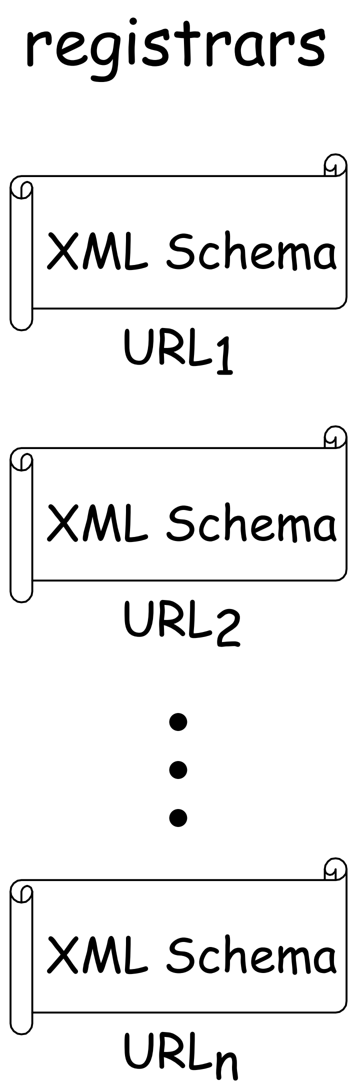
- Registry of metadata formats for OpenURL
  - http://www.sfxit.com/openurl/
  - http://lib-www.lanl.gov/~herbertv/papers/icpp02-draft.pdf
- other uses?

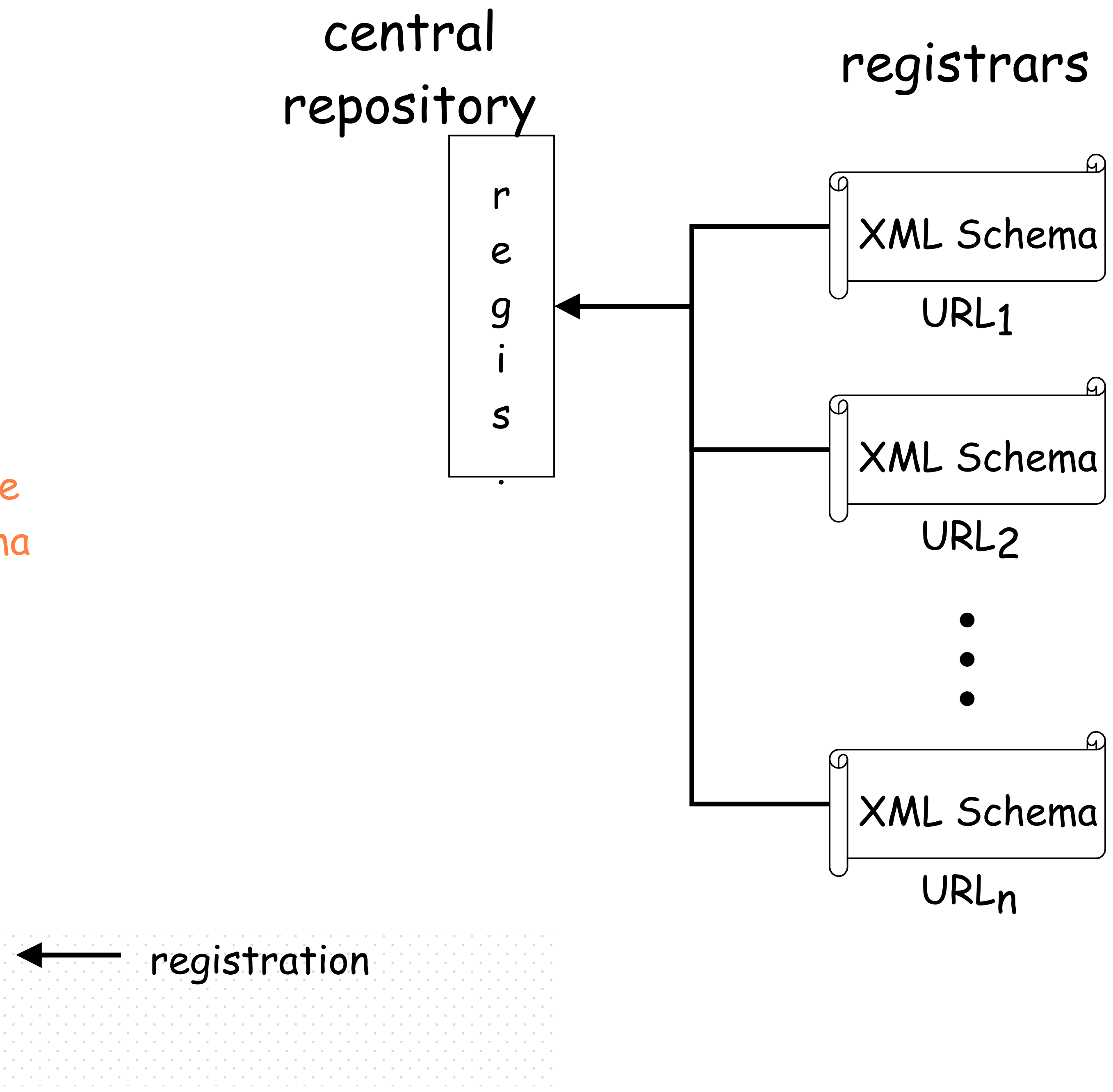
registry model



Goal:

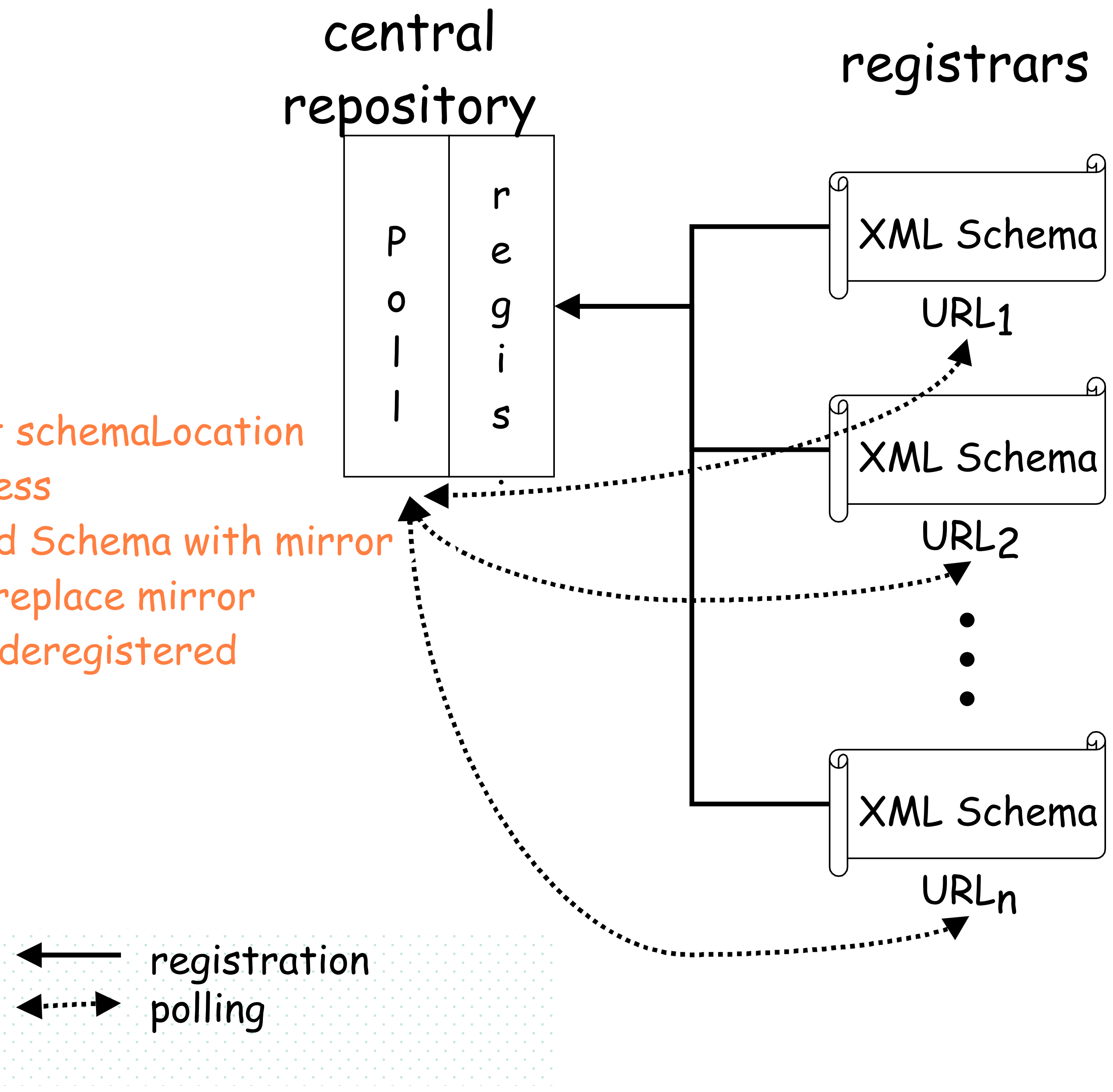
- inform linking servers re Schema
- ease of admin for all parties involved
- limit human overhead




central repository
registrars
r e g i s .
XML Schema
URL1
XML Schema
URL2
XML Schema
URLn
registration
Registry:
• schemaLocation
• registration date
• mirror of Schema

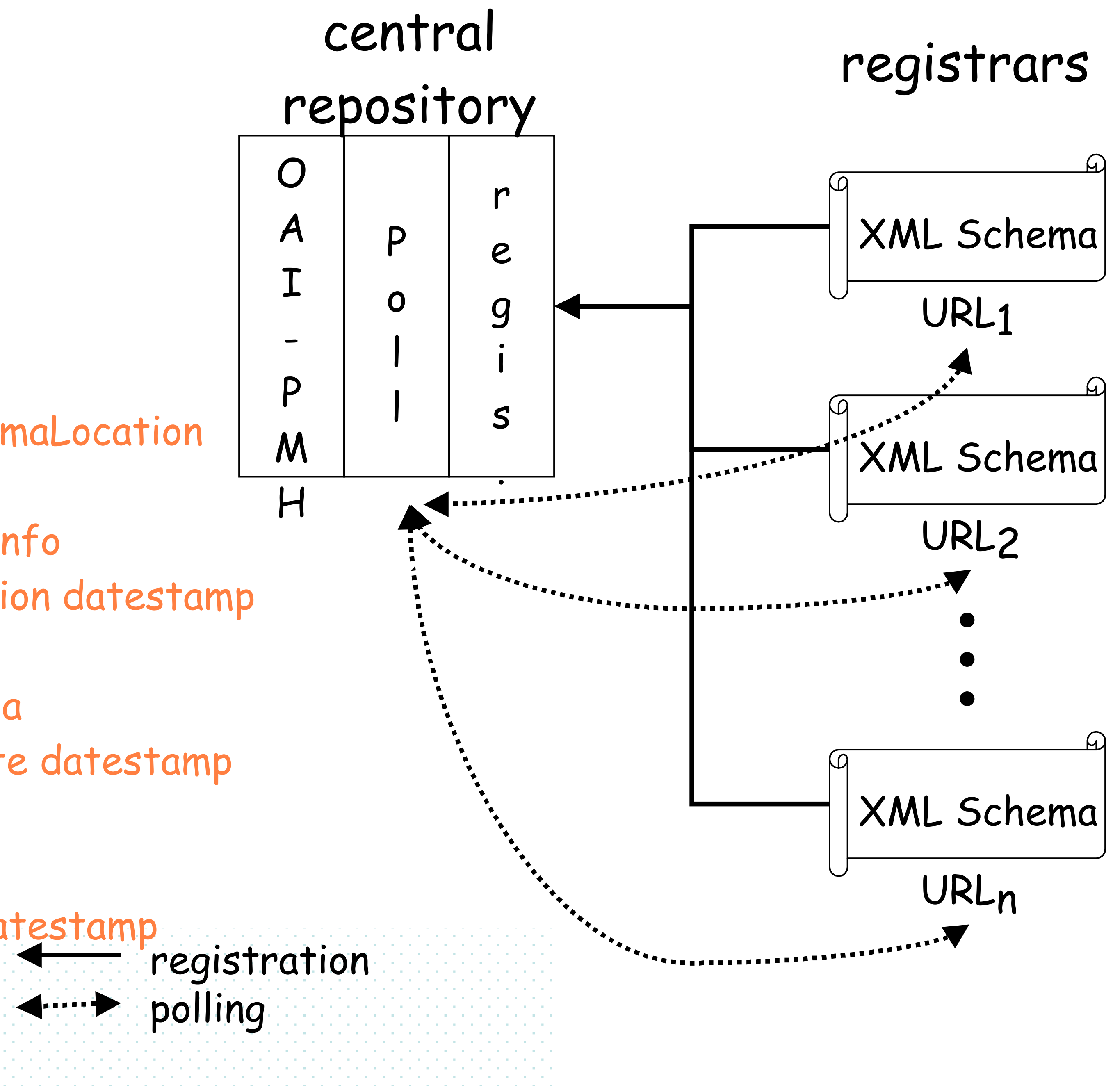
Poll:

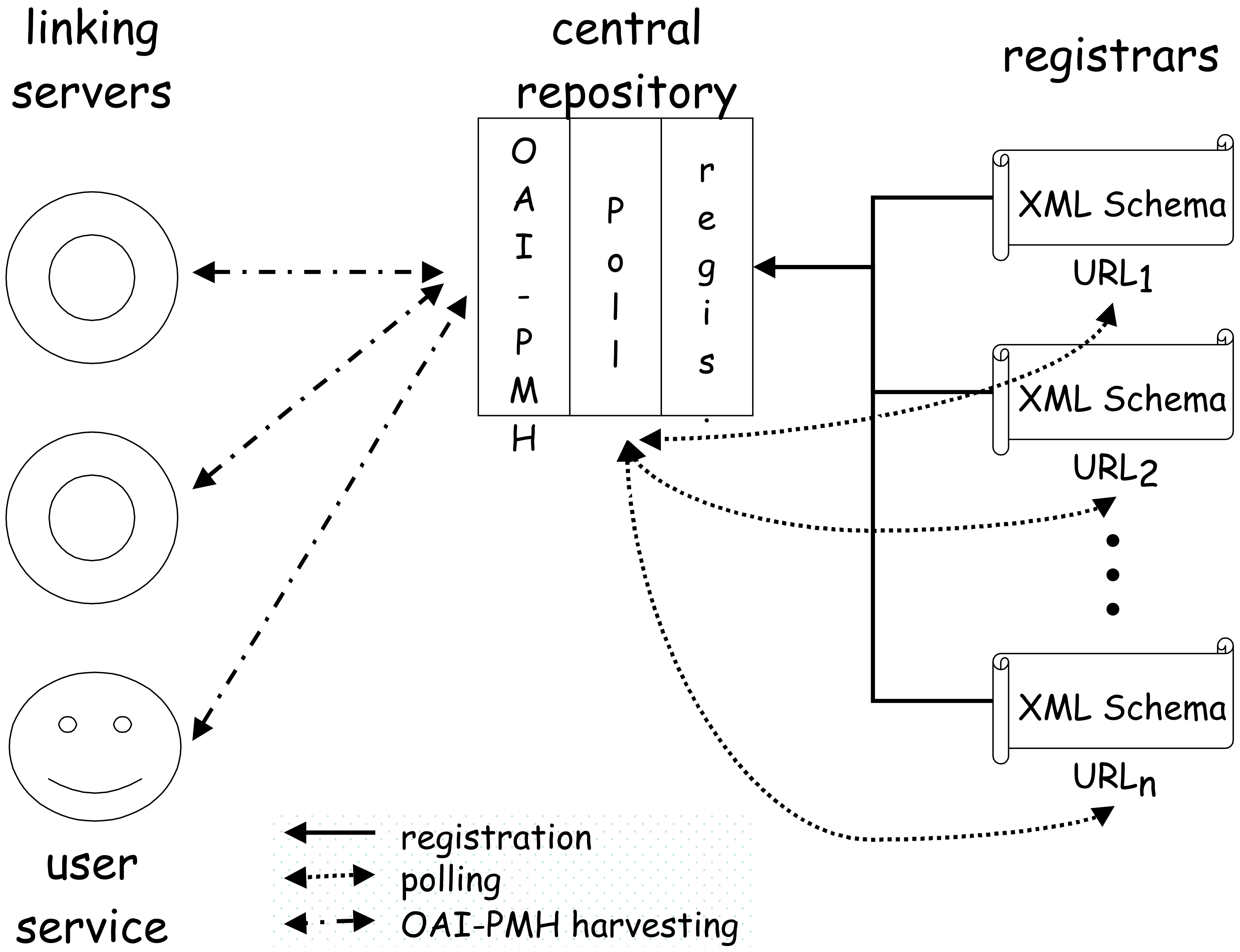
- fetch schema at schemaLocation
- log failure/success
- compare fetched Schema with mirror
    - changed => replace mirror
    - removed => deregistered



OAI repo:

- record-ids = schemaLocation
- oai_dc record :
    - registration info
    - (de)registration datestamp
- xsi record :
    - mirror schema
    - schema update datestamp
- poll record :
    - process info
    - recent poll datestamp




linking servers
central repository
registrars
OAI-PMH
Poll
regis
XML Schema
URL1
XML Schema
URL2
XML Schema
URLn
user service
registration
polling
OAI-PMH harvesting