Repository Deposit Service Description

OR 2007: the 2nd International Conference on Open Repositories
San Antonio, Texas, USA, 23-26 Jan 2007

Presenter: Julie Allinson, UKOLN, University of Bath
Co-authors: Rachel Heery (UKOLN), Martin Morrey (Intrallect), Christopher Gutteridge (Southampton), and Jim Downing (Cambridge)
Overview

• Background and context
• Requirements for depositing content in repositories
• Defining a lightweight deposit service
• Developing the service
• Proof-of-concept implementation - update
Context

- Higher (and Further) Education in the United Kingdom
- JISC – the Joint Information Systems Committee
- JISC – considerable investment in UK repositories R&D over the last 5 years, and continuing
  - FAIR Programme (2002-2005)
  - Exchange for Learning (X4L) Programme (2002-2005)
  - Digital Repositories Programme (2005-2007)
Deposit API

- Deposit API activity was brought together to find lightweight solution to assist populating repositories within timescales of JISC programmes.
- It comprised a group of repository software developers from Eprints.org, DSpace, Fedora, Intrallect and others.
- Facilitated by the JISC Repositories Research Team.
- To address the need for a common Deposit standard.
Background motivation

• In general, developers are not creating repository systems and software from scratch
• Repositories must interface with each other, with users and with other applications within institutions and the wider information landscape
  – VLEs, authoring tools, packaging tools, name authority services, classification services and research systems
• There is no common deposit API or protocol
Pain points

• no standardised way of transferring existing collections of digital objects and/or metadata from a filesystem or legacy database into a repository
• no standard interface for tagging, packaging or authoring tools to upload catalogued objects into a repository
• no standard interface for transferring digital objects between repositories
• no way of initiating a contribution workflow from outside a repository system
• no way of including deposit into a repository a part of service orientated architecture

for harvesting there is OAI-PMH – this has had a major impact

There is no equivalent mechanism for deposit
Why is deposit so important?

- Without it, there is nothing in our repositories
- Ensuring the emerging network of repositories is well populated with content is a PRIORITY
- Encouraging deposit is one of the most difficult cultural issues for repositories
- Technology needs to support culture change and advocacy, through
  - ease of use
  - multiple deposit
  - auto-deposit
  - NOT closed or proprietary mechanisms
Repository

Stores, manages and makes available content and metadata
• Deposit interface
• Delete interface
• Search interface
• Harvest interface
• Obtain interface

From Andy Powell, A 'service oriented' view of the JISC Information Environment:

• Similarly, the ORE initiative identifies put (deposit), obtain and harvest services
Deposit – abstract service definition

• A Deposit interface: *Provides an interface through which content and metadata can be deposited and initiates ingest process for local storage.*

  Summarised from Andy Powell, A 'service oriented' view of the JISC Information Environment:

• Put: *A put service supports the request for ingest of one or more surrogates into a repository, thereby allowing the addition of digital objects to the repositories’ collection*

  From An interoperable fabric for scholarly value chains:
  http://www.dlib.org/dlib/october06/vandesompel/10vandesompel.html
A note on terminology

• **Add** - used by the e-Framework
• **Deposit** - terminology of choice for this work
• **Put** - used by ORE
• **Submit**
• **Post** - used for blogs

**Ingest**
- deposit, put, add etc. may be part of an ingest process, along with other functions
- may include both automated and manual procedures including format checking, editorial control, quality assurance mechanisms, etc.
- defined by OAIS
- these are out of scope for this activity

Largely synonymous, with subtle differences, often related to community of use
User requirements / scenarios

1. Author deposits using a desktop authoring system to a mediated multiple deposit service
2. A user submits an IMS-compliant learning object to a National Repository using a client application
3. Deposit into multiple repositories
4. Transfer between intermediate hosts
5. Repositories share improved metadata
6. Experimental data output from spectrometer is 'saved as' a file and a file containing metadata on operational parameters is also generated. A data capture service is invoked and the files pertaining to the experiment are deposited, along with the necessary metadata, in the laboratory repository.

See more at http://www.ukoln.ac.uk/repositories/digirep/
Scenario 1: Author deposits using a desktop authoring system to a mediated multiple deposit service

Author A deposits via an easy-deposit desktop application into the institutional repository's mediated deposit queue.

Librarian L completes the deposit through the repository interface.

A lightweight deposit web service can facilitate this transfer of object(s).

Librarian L invokes deposit of a surrogate into arxiv.org.
Scenario 2: A user submits an IMS-compliant learning object to a National Repository using a client application

A user wishes to submit an IMS-compliant content package to a repository using a client application.

The user can choose from a list of ‘groups/collections’ to which they are allowed to deposit, in this centralised national LO repository. They are not required to use the repository interface, but can deposit via a decentralised client.

A lightweight deposit web service can facilitate this transfer of object(s).
Scenario 3: Deposit in multiple repositories

A depositor is required to submit to a Research Council repository, but they also wish to deposit into their institutional repository and a relevant subject repository.

The depositor can choose one or more repositories to deposit into.

A lightweight deposit web service can facilitate this transfer of object(s).
Scenario 4: transfer between intermediate hosts

A repository may transfer objects to other repositories, or services, e.g. a preservation service.

A lightweight deposit web service can facilitate this transfer of object(s).

Subsequent repositories may also transfer objects.
Scenario 5: Repositories share improved metadata (put both ways)

Repository A deposits an object in another repository.

A lightweight deposit web service can facilitate this transfer of object(s).

Repository B improves the metadata and deposits the object back into repository A.
Scenario 6: laboratory auto-deposit

Experimental data output from laboratory machines is deposited, along with the necessary metadata, in the laboratory repository in an automated process.

A lightweight deposit web service can facilitate this transfer of object(s).

A metadata record is also deposited into the Institutional Repository.
Some functional requirements

A Deposit service should:

• be generic enough to support wide range of heterogeneous repositories
  – scholarly publications, data, learning objects, images, etc.
• accept submission of different digital object types in consistent way:
  – data and/or metadata in the form of complex objects or content packages
• support different workflows for deposit, e.g.
  – user to multiple repositories via intermediate client
  – user to repository, repository to additional repositories
  – user-triggered and machine-triggered deposit
• accept large-scale (scientific datasets)
• support statuses, e.g. deposit to different states of a workflow
• support collections and changes in policy and permissions
• support differences in repository policy
• support non-instantaneous processes, e.g. deposit pending mediation
• support validation report and integrity checks
• support anonymous deposit
• support more complex, authenticated deposit
• support acceptance and handling of incomplete records
• support rejection of records (reasons for rejection are out of scope)
• support human-selected targets for deposit
• support different deposit requests
Defining a lightweight deposit service

• Define abstract service scope
  – information models and APIs must be developed in manner neutral to implementation binding

  **Abstract service:** a discrete piece of technical functionality required to fulfil a specific requirement or set of requirements
  
  Synonymous with a ‘service genre’ in the JISC DEST e-Framework

• Examine existing protocols and specifications
  – could they be used implement the defined abstract service?

• Evaluate and decide whether a new protocol or API is required
Deposit – abstract service definition

• A Deposit interface: *Provides an interface through which content and metadata can be deposited and initiates ingest process for local storage.*


• Put: *A put service supports the request for ingest of one or more surrogates into a repository, thereby allowing the addition of digital objects to the repositories’ collection*

  From An interoperable fabric for scholarly value chains: http://www.dlib.org/dlib/october06/vandesompel/10vandesompel.html
Existing standards

- WebDAV (http://www.webdav.org/)
- JSR 170 (http://www.jcp.org/en/jsr/detail?id=170)
- SRW Update (http://www.loc.gov/standards/sru/)
- Flickr Deposit API (http://www.flickr.com/services/api/)
- Fedora Deposit API (http://www.fedora.info/definitions/1/0/api/)
- OKI OSID (http://www.okiproject.org/)
- ECL (http://ecl.iat.sfu.ca/)
Deposit – two components

- **Deposit**: service offered by a repository, allowing remote users (machines or people) to upload data
  - data in:
    - deposit request with optional parameters (e.g. digital object ‘semantics’, metadata formats..)
  - data out:
    - status (success, failure, pending), receipt confirmation and digital object identifier
- **Explain**: service offered by a repository, allowing remote users (machines or people) to inspect the repository for policy and/or other data
  - data in:
    - introspection request (“explain”)
  - data out:
    - introspection response (“repository policy info”)
Draft XML serialisations

Deposit API explain serialisation

This page is part of Deposit API.

```
<response>
  ...
</response>
```

```
<responseCode/>
<responseMessage/>
<!-- If response code is success: -- -->
<repository>
  <globals>
    <repositoryId>[CDATA]</repositoryId>
    <!-- Description -->
    <policies/>
  </global>
  <!-- Same schema as //explain/repository/collections/collection -->
  <defaultCollection>
  </defaultCollection>

  <!-- For depth >0 requests -->
  <collections>
    <collection>
      <id>[CDATA]<id>
      <description>[CDATA]<description>
      <displayURL>[URL]<display>
      <acceptedFormats>
        <!-- Contains uri & description -->
        *<format/>
      </acceptedFormats>
  </defaultCollection>
```

http://www.ukoln.ac.uk/repositories/digirep
Deposit service specification

- To recap, following the scope defined earlier
- the repository developers came up two services: deposit and explain
- and a draft XML serialisation for each
- the service will work by the client issuing XML commands over HTTP to the repository Deposit service
- the service responds with formatted XML messages
- other approaches may also be considered, e.g. SOAP
- a layered approach was taken, with the specification of two levels of compliance.
  - Level 0 compliance requires a set of mandatory elements
  - Level 1 offers a set of additional optional elements that may or may not be used
Some issues

• **Boundaries between deposit and ingest**
  – what has already happened at point of deposit? regarding metadata and identifiers
  – how far does the deposit service need to validate what is being deposited
  – and can it reject deposit requests?

• **Data integrity**
  – is there requirement to get back (export) exact object that was deposited?

• **Multiple data types, metadata formats and content packages**
  – how far should the deposit service check its ability to accept what is being deposited?
  – Can look up of policy rules be done as a request to service registry?
  – how far is look up of policy rules automated?

• **Authorisation and authentication**
  – how will the deposit service check the authority of the person/machine doing the ‘putting’
  – how will it interface with auth services?
Next steps

**Finish it and test it!**

- At the moment, the deposit web service is still embryonic
- To take it forward, a funding proposal has been submitted to finalise the original deposit API work
- Test it against different repository software
  - Eprints
  - DSpace
  - Fedora
  - Intrallect intraLibrary
- Build a client implementation
- Iteratively revise and re-test
- Disseminate and embed into the repositories community
Final thoughts ...

- This work is aligned with the vision of the JISC-DEST E-Framework and the soa approach in general
- also with the JISC Information Environment commitment to interoperability and the use of web services to facilitate interaction between Repositories and other services
- and with the objectives of the Object Re-use and Exchange Initiative and the definition of a ‘put’ interface

Thank you …