Simple Web service Offering Repository Deposit (SWORD)

Project kick-off meeting
Birkbeck College, London, 30th April 2007

Julie Allinson, UKOLN, University of Bath

www.ukoln.ac.uk/repositories/digirep/index/SWORD

<j.allinson@ukoln.ac.uk>

UKOLN is supported by:

JISC

MLA MUSEUMS LIBRARIES ARCHIVES COUNCIL

UNIVERSITY OF BATH

www.bath.ac.uk

UKOLN www.ukoln.ac.uk

a centre of expertise in digital information management

Project plan : aims

- To improve the efficiency and quality of the repository ‘Ingest’ function
- To diversify and expedite the options for timely population of repositories with content
- To facilitate the creation and use of common deposit interfaces
- To improve repository interoperability as outlined in the Information Environment
- To take a service-oriented approach to development as outlined by the E-Framework

Project plan : objectives

- To produce a standard mechanism for depositing content in repositories
- To test and refine the lightweight protocol originally formulated by a small group working within the Digital Repositories Programme (the Deposit API)
- To evaluate existing standards that might be used to offer a deposit web service
- To implement the deposit service in EPrints, DSpace, Fedora and IntraLibrary
- To develop a prototype ‘smart deposit’ tool
- To disseminate the resulting work and encourage community uptake
- To ensure that the approach developed by this project is cognisant of UK requirements (as defined by the JISC Common Repository Interfaces Group – CRIG) and International work in this area (including the OAI-ORE activity)

Workpackages

- 1: Documentation of the deposit API
  - March to July
- 2: Technical development
  - April to August
- 3: User testing and feedback
  - June to August
- 4: Community acceptance and dissemination
  - March to August
- 5: Project management
  - March to August

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
- Two meetings: March 2006, July 2006
  - Discussion of scenarios/use cases; Requirements; Draft XML serialisations

User requirements / scenarios

- Author deposits using a desktop authoring system to a mediated multiple deposit service
- A user submits an IMS-compliant learning object to a National Repository using a client application
- Deposit into multiple repositories
- Transfer between intermediate hosts
- Repositories share improved metadata
- 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.

From at http://www.ukoln.ac.uk/repositories/digirep/

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

Scope

- In
  - Deposit
  - Permissions and conditions for deposit
- Out
  - Update and delete
  - Mappings between metadata/packaging formats
  - Identifier solutions
  - Relationships between digital objects
  - Tracking and provenance
  - Authentication
  - …

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

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?

Existing standards

- WebDAV (http://www.webdav.org/)
- JSR 170 (http://www.jcp.org/en/jsr/detail?id=170)
- JSR 283 (http://www.jcp.org/en/jsr/detail?id=283)
- 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/)
- ATOM Publishing Protocol (http://www.ietf.org/html-charters/atompub-charter.h)

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:
http://www.ukoln.ac.uk/distributed-systems/jisc-ie/arch/soa/jisc-ie-soa.pdf

- 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

Deposit – two components

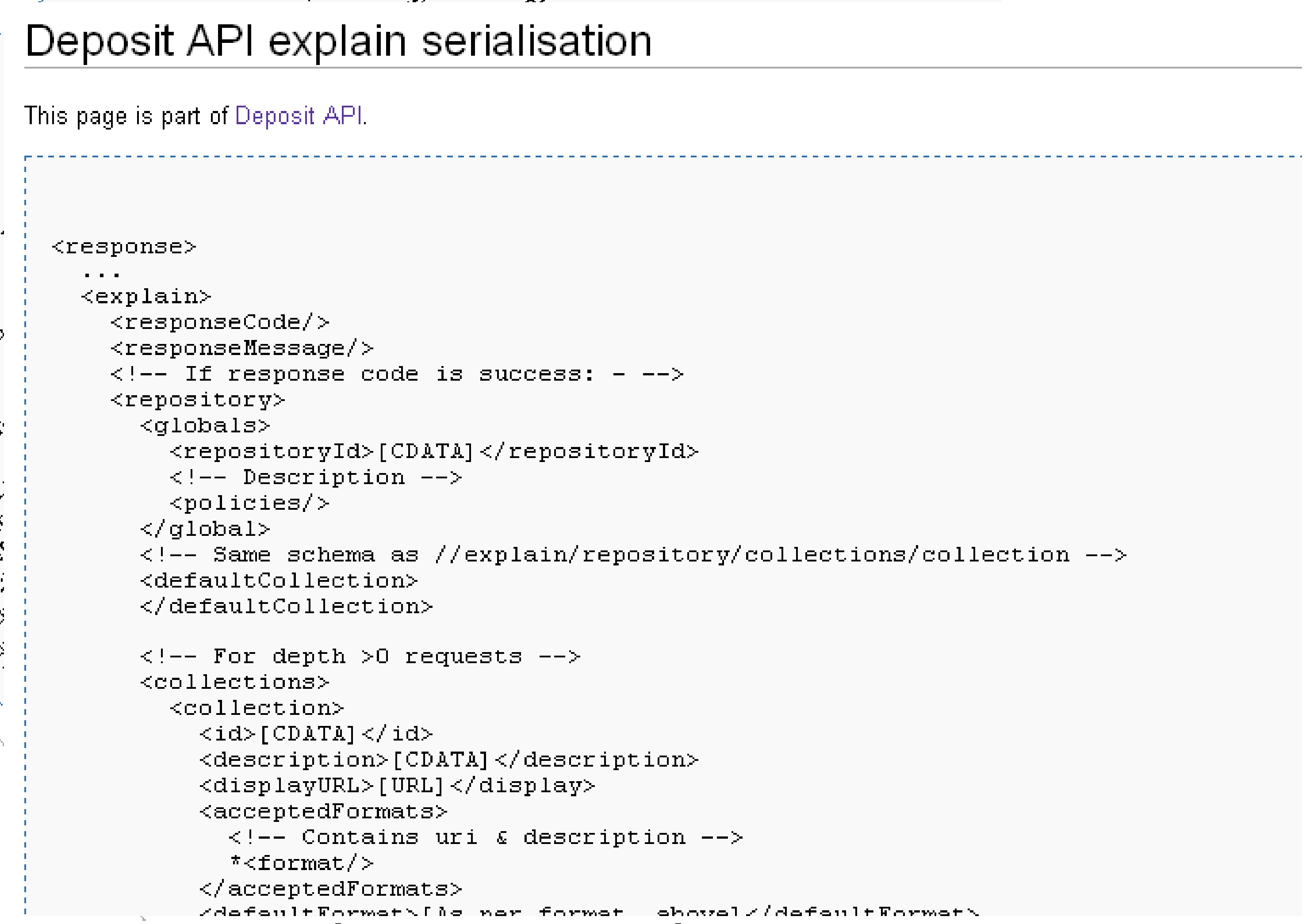
- **Deposit**: service offered by a repository, allowing remote users (machines or people) to upload data
  - data in:
    - deposit request with optional parameters (e.g.digital object ‘semantics’, metadata formats..)
  - data out:
    - status (success, failure, pending), receipt confirmation and digital 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

eposit API deposit serialisation

```
<response>
  ...
  <deposit>
    <!--  I suggest
          is a sub
    -->
    <responseCo
    <!--  Th
    -->
    <err
    content
    parse
    chec
    unl
    b
```

using the data in the receip
response obj
these way given.


Deposit API explain serialisation

This page is part of Deposit API.

```
<response>
  ...
  <explain>
    <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>
          <defaultFormat>[As per format, above]</defaultFormat>
```

http://www.ukoln.ac.uk/repositories/digirep

Deposit service specification

- 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, with the specification of two levels of compliance at the moment.
  - 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

The work …

- define the service
  - Implementation-neutral information models
- 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, or 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