SWORD

Simple Web-service Offering Repository Deposit

Defining Image Access final project meeting

Wolfson College, Oxford, 22nd June 2007

Julie Allinson <j.allinson@ukoln.ac.uk>

Repositories Research Officer

UKOLN

UKOLN is supported by:

JISC

Supported by MLA Museums, Libraries and Archives Council

www.ukoln.ac.uk

A centre of expertise in digital information management

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

UNIVERSITY OF BATH

SWORD

- Aims to:
  - Improve efficiency of the repository 'Ingest' function
  - Improve options for populating (multiple) repositories with content
  - Support common deposit interfaces
  - Achieve repository interoperability
- Through:
  - A standard specification for depositing content in repositories
  - Implemented and tested (and refined) in EPrints, DSpace, Fedora and IntraLibrary,
  - and a prototype 'smart deposit' tool
- At all times being 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)

SWORD and Deposit API

- SWORD partners are
  - UKOLN, University of Bath
  - University of Southampton (EPrints)
  - University of Aberystwyth (DSpace, Fedora, reference client)
  - Intrallect (IntraLibrary)
- Taking forward the results of the Deposit API activity
  - a group of repository software developers from Eprints.org, DSpace, Fedora, Intrallect and others
  - facilitated by the JISC Repositories Research Team
  - met to address the need for a common Deposit standard
  - discussed scenarios/use cases; requirements; draft XML serialisations

User requirements / scenarios

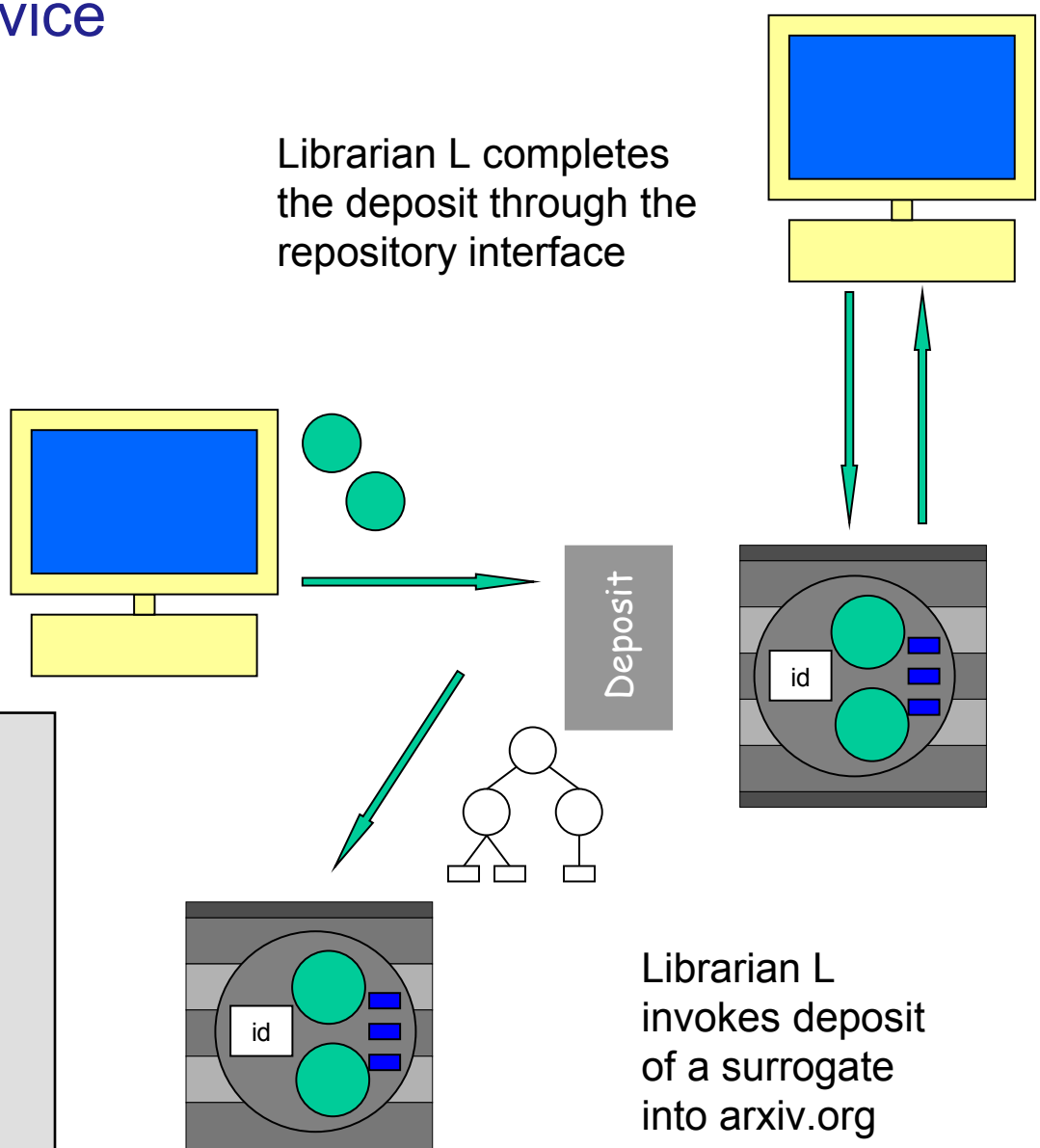
- Author deposits using a desktop authoring system to a mediated multiple deposit service
- A user submits an IMS-compliant learning object to a National Repository using a client application
- Deposit into multiple repositories
- Transfer between intermediate hosts
- Repositories share improved metadata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

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

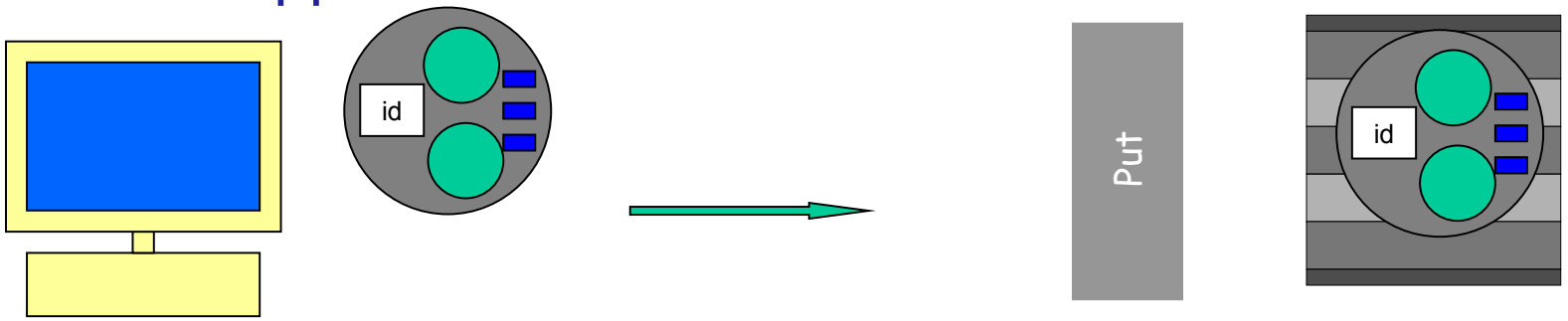
Librarian L completes the deposit through the repository interface



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

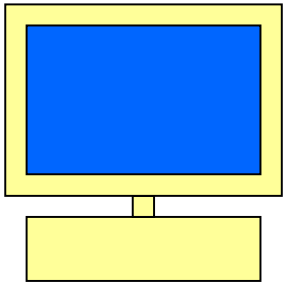


A user wishes to submit an IMS-compliant content package to a repository using a client application

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

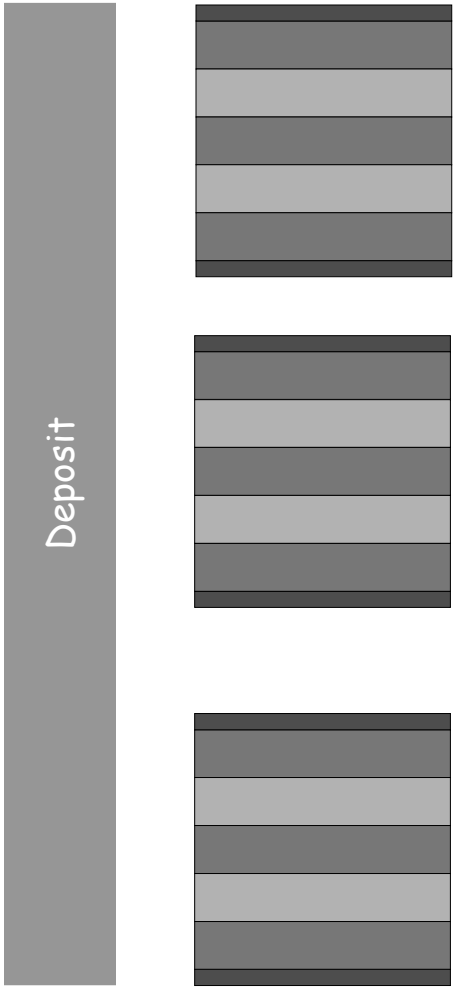
The user can choose from a list of 'groups/collections' to which they are allowed to deposit, in this centralised national LO repository. They are not required to use the repository interface, but can deposit via a decentralised client.

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)

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

Deposit – two components

- **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”)
- **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 identifier

Deposit - parameters

- Mandatory (level 0 compliance)
  - deposit any type of content
  - repository or collection id
  - identifier
  - deposit status (accepted, rejected, error), error codes, error description
  - treatment description
- Optional (mandatory for level 1 compliance)
  - mediated deposit
  - repository / collection name
  - collection policy, description
  - accepted formats
  - format namespace
  - source repository
  - checksum
  - compliance level
  - additional identifiers

Existing standards considered

- 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)

Atom Publishing Protocol

- "the **Atom Publishing Protocol** is an application-level **protocol** for **publishing** and editing Web resources"
- Benefits of using the Atom Publishing Protocol
  - Supports many of our parameters and requirements, in particular file deposit
  - It already exists and has an active development community
  - Support is growing
  - It is well-used in popular applications
  - It has an extension mechanism
  - Google have created their own profile (gdata)
- Drawbacks / risks
  - This isn't what it was designed for – are we attempting to fit our square requirements into round holes?
  - Without significant 'interpretation', it is only possible to deposit a single package/file OR an atom document – this means that we need to package up metadata and files

APP and SWORD parameters

- **Mandatory** (level 0 compliance)
  - deposit any type of content – APP yes
  - repository or collection id – APP yes
  - Identifier – APP yes
  - deposit status (accepted, rejected, error), error codes, error description – APP yes (and extension)
  - treatment description - extension
  - deposit id
  - target collection
- **Optional** (mandatory for level 1 compliance)
  - mediated deposit support - extension
  - on-behalf-of target user - extension
  - repository / collection name - APP yes
  - collection policy, description - extension
  - accepted formats – APP yes
  - format namespace - extension
  - source repository – APP yes
  - checksum - extension
  - compliance level - extension
  - additional identifiers – APP yes

What is SWORD doing …

- Agree scope (this is a small project, there is a lot which is out of scope)
- Agree a protocol
  - Atom Publishing Protocol (APP)
  - SWORD profile of APP
- 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