SWORD

Simple Web-service Offering Repository Deposit

JISC CETIS SIG meeting
University of Strathclyde, Glasgow
Friday 29th June 2007

Julie Allinson
UKOLN,
University of Bath

Image: www.flickr.com/photos/venegas/8519843/

UKOLN

The order of things

- ending at the beginning …
- before SWORD - background and context
- SWORD – the project
- why SWORD?
- where we are at
  - scope, requirements and parameters
  - defining the service
  - specifications
- where we are going
  - proof-of-concept

Before SWORD there was Deposit API

- Deposit API activity
- facilitated by the JISC Repositories Research Team in 2006
- a group of repository software developers from Eprints.org, DSpace, Fedora, Intrallect and others
- brought together to address the requirement for a common deposit standard
- for populating the growing network of repositories with content
- within the timescales of JISC repositories programmes

Broadly motivated by …

- in general, developers are not creating repository systems and software from scratch
- repositories must interface with …
  - each other
  - users
  - other applications within the institution
  - services in the wider information landscape
  - (such as VLEs, authoring tools, packaging tools, name authority services, classification services and research systems)
- there is no common deposit API or protocol
- (OAI-PMH has made metadata interoperability an imperfect reality)

A note on terminology

- Add - used by the e-Framework
- Deposit - our terminology of choice
- Put - formerly used by ORE
- Register - now used by ORE
- Submit - use for generic'forms'
- Post - used for blogs

broadly synonymous, with *subtle* differences, often related to community of use

- **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

Pain points - what can't we do?

- 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

From pain to possibility - some scenarios

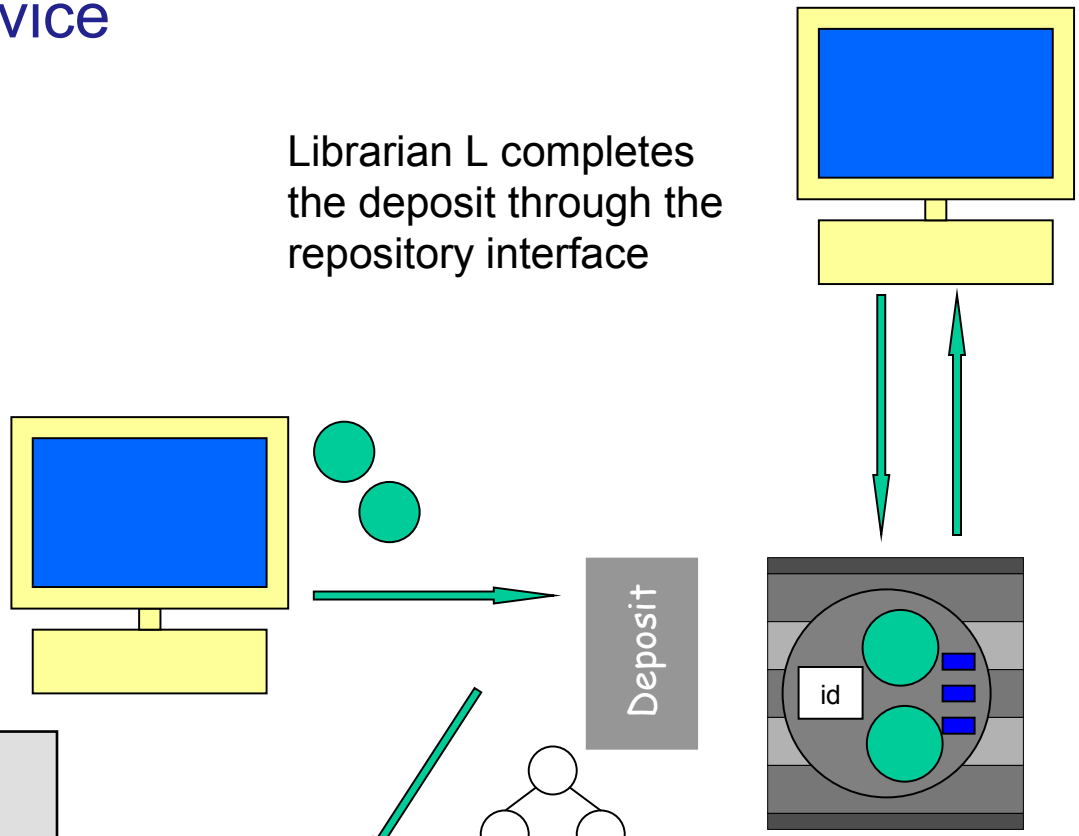
- Author deposits using a desktop authoring system to a mediated multiple deposit service
- A user submits an IMS-compliant learning object to a National Repository using a client application
- Deposit into multiple repositories
- Transfer between intermediate hosts
- Repositories share improved metadata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

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

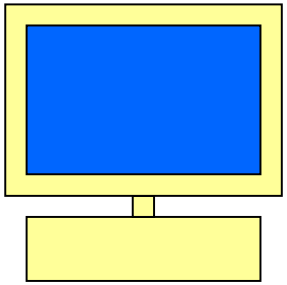
Librarian L completes the deposit through the repository interface



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

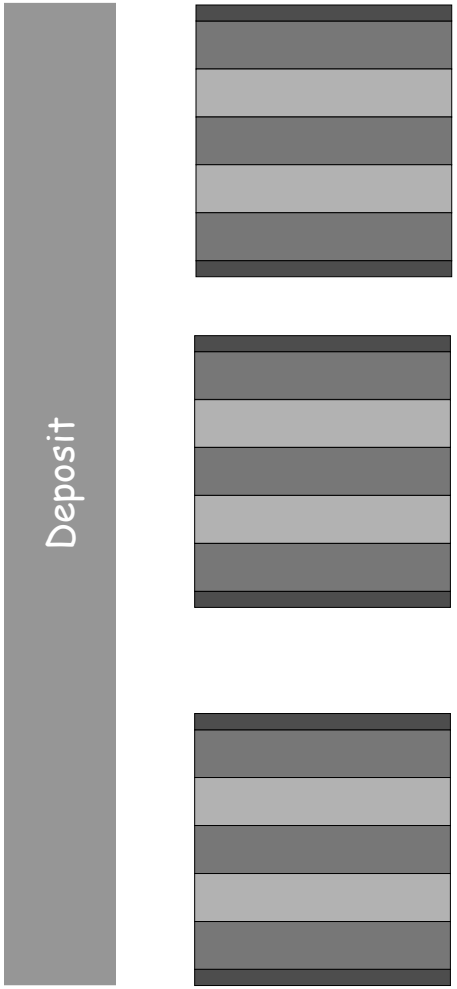
Librarian L invokes deposit of a surrogate into arxiv.org

Scenario 3 : Deposit in multiple repositories



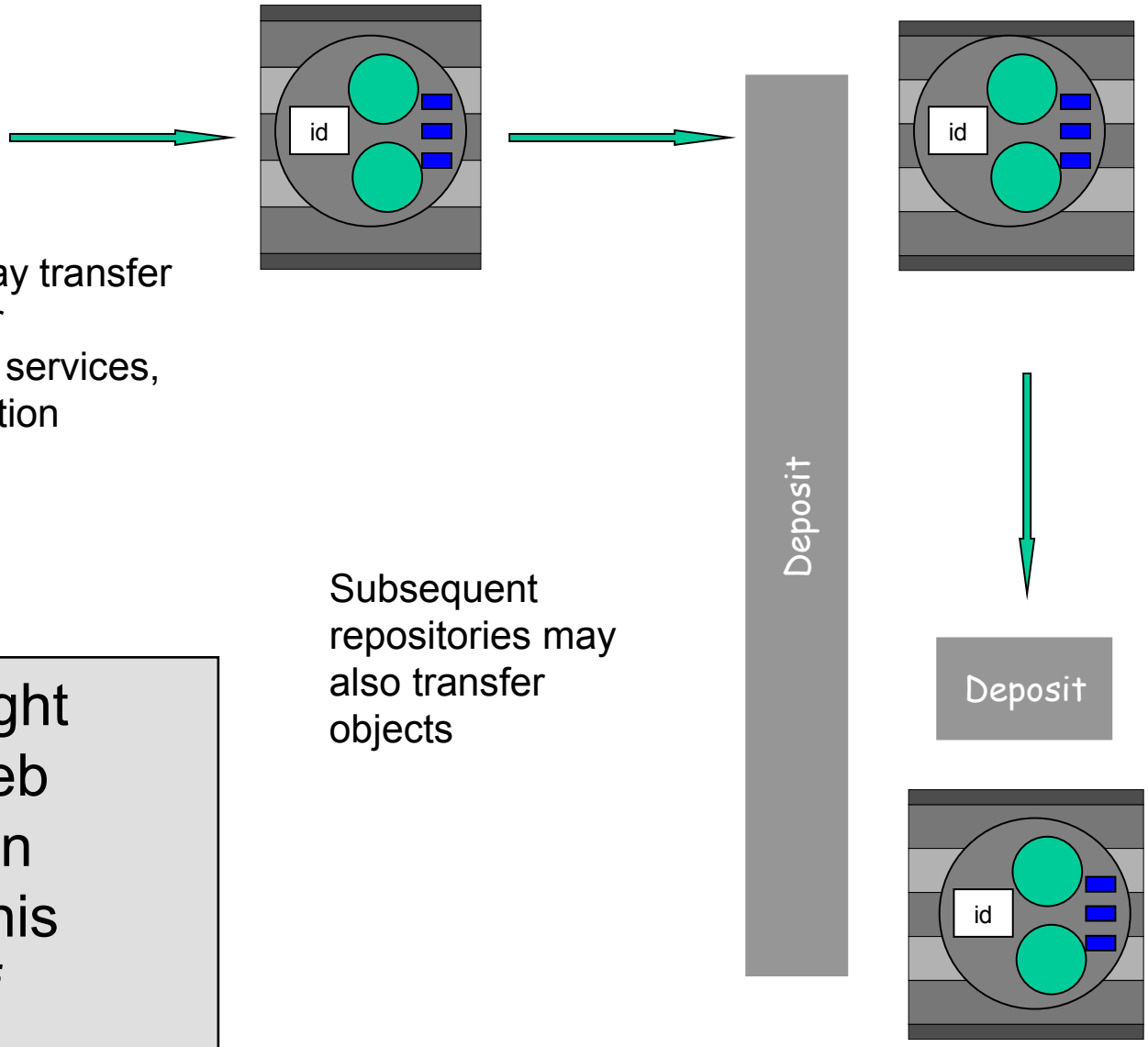
A depositor is required to submit to a Research Council repository, but they also wish to deposit into their institutional repository and a relevant subject repository

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

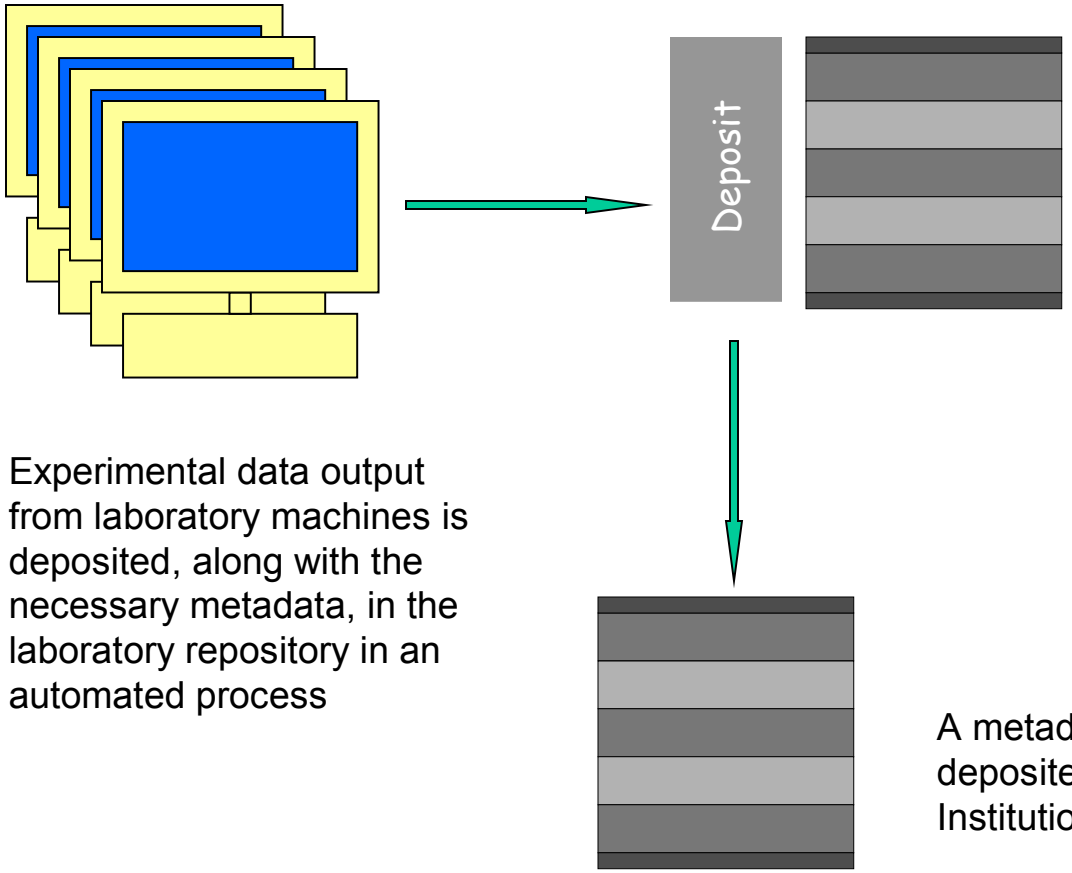


A repository may transfer objects to other repositories, or services, e.g. a preservation service

Subsequent repositories may also transfer objects

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

Scenario 6 : laboratory auto-deposit



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

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

A metadata record is also deposited into the Institutional Repository

And so to SWORD

- SWORD:
  - Simple
  - Web-service
  - (Offering)
  - Repository
  - Deposit
- SWORD partners are
  - UKOLN, University of Bath
  - University of Southampton (EPrints)
  - University of Aberystwyth (DSpace, Fedora, reference client)
  - Intrallect (IntraLibrary)
- 6 months, started March 2007
- building on the Deposit API work

SWORD – aims and objectives

- 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)

Steps towards offering a deposit service

- explore use cases – Deposit API
- identify requirements – Deposit API
- parameters – outlined by Deposit API - refined by SWORD
- define services: deposit and explain – outlined by Deposit API – refined by SWORD
- agree layered approach – outlined by Deposit API - refined by SWORD
- draft XML serialisation to better understand requirements – Deposit API
- reviewed existing standards – SWORD
- agree standard – SWORD
- implement and test - SWORD

Functional requirements (a select few)

- 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 collections and changes in policy and permissions
- support non-instantaneous processes, e.g. deposit pending mediation
- support more complex, authenticated deposit

Deposit – turning requirements into 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

Defining the deposit service(s)

- **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

Future proofing with layers

- layered approach
- two levels of compliance
  - Level 0 compliance requires a set of mandatory elements
    - and a set of optional elements
  - Level 1 offers a set of additional elements for richer functionality
    - mandatory at level 1

Reviewing 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 http://www.ietf.org/html.charters/atompub-charter.html)

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)
  - extension - mediated deposit support
  - extension - on-behalf-of target user
  - APP yes - repository / collection name
  - extension - collection policy, description
  - APP yes - accepted formats
  - extension - format namespace
  - APP yes - source repository
  - extension - checksum
  - extension - compliance level
  - APP yes - additional identifiers

A bit more detail about APP and the SWORD profile

- APP works by issuing HTTP requests (GET, POST)
- HTTP response and APP or ATOM document is returned
- HTTP header extensions
  - Content-MD5
  - X-On-Behalf-Of
  - X-Deposit-ID
  - X-Error-Code
  - X-Format-Namespace
- APP / ATOM extensions
  - <dcterms:accrualPolicy>
  - <dcterms:abstract>
  - <sword:mediation>
  - <sword:defaultCollection>
  - <sword:treatment>
  - <sword:formatNamespace>
  - <sword:level>

Example : ‘explain’

```
GET /app/servicedocument HTTP/1.1
Host: www.myrepository.ac.uk
X-On-Behalf-Of: lcarr
```

```
<?xml version="1.0" encoding='utf-8'?>
<service xmlns="http://www.w3.org/2007/app#"
         xmlns:atom="http://www.w3.org/2005/Atom"
         xmlns:sword="http://purl.org/sword/"
         xmlns:dcterms="http://purl.org/dc/terms/">
 <sword:level>1</sword:level>
 <sword:verbose>true</sword:verbose>
 <sword:noOp>true</sword:noOp>
 <workspace>
   <atom:title>Main Site</atom:title>
   <collection
       href="http://www.myrepository.ac.uk/atom/geography-collection" >
     <atom:title>My Repository : Geography Collection</atom:title>
     <app:accept>application/xml, application/zip, application/atom+xml</app:accept>
     <dcterms:accrualPolicy>Collection Policy</dcterms:accrualPolicy>
     <dcterms:abstract>Collection description</dcterms:abstract>
     <sword:mediation>true</sword:mediation>
     <sword:defaultCollection>true</sword:defaultcollection>
     <sword:treatment>Treatment description</sword:treatment>
     <sword:namespace>uri</sword:namespace>
   </collection>
 </workspace>
</service>
```

Example – ‘deposit’ with HTTP POST

```
POST /geography-collection HTTP/1.1
Host: www.myrepository.ac.uk/app
Content-Type: application/zip
Slug: My Deposit
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: nnn
Content-MD5: md5-digest
X-On-Behalf-Of: lcarr
X-Verbose: true
X-No-Op: true
X-Format-Namespace: agreed-format-namespace
X-Deposit-ID: deposit-id
```

Example : ‘receipt’

```
HTTP/1.1 201 Created
Date: Mon, 14 May 2007 14:27:11 GMT
Content-Length: nnn
Content-Type: application/atom+xml; charset="utf-8"
Location: http://www.myrepository.ac.uk/collection/edit/my_deposit.atom
```

```
<?xml version="1.0"?>
 <entry xmlns:atom="http://www.w3.org/2005/Atom"
        xmlns:sword="http://purl.org/sword/">
   <title>My Deposit</title>
   <id>deposit-id</id>
   <updated>2007-05-14T14:27:08Z</updated>
   <author><name>Les Carr</name></author>
   <summary type="text" />
   <content type="application/zip"
      src="http://www.myrepository.ac.uk/my_deposit.zip"/>
   <link rel="edit-media"
      href="http://www.myrepository.ac.uk/lcarr/workflow/my_deposit" />
   <link rel="edit"
      href="http://www.myrepository.ac.uk/lcarr/workflow/my_deposit.atom" />
   <contributor><name>Martin Morrey</name></contributor>
   <source>
        <generator>Repository id</generator>
   </source>
   <sword:treatment>Treatment description</sword:treatment>
</entry>
```

Scope and assumptions

- authorisation and authentication
  - too big an issue for SWORD
  - APP mandates minimum HTTP authentication
  - other authentication mechanisms are left to the implementer
- deposit and ingest
  - we assume that metadata already exists
  - identifiers are an issue – to what extent should pre-existing identifiers and those created during the deposit be maintained
  - how far does the deposit service need to validate the deposit
- data integrity and provenance
  - no ***requirement*** to get back the exact package deposited?
- data types, metadata formats and content packages
  - how far should the deposit service check its ability to accept what is being deposited?
  - is there any value in being able to deposit a package format that the accepting repository doesn't support?
    - SWORD is testing a minimal set of packaging formats, mappings and translations – out of scope?
    - Extracting metadata from packages – a post deposit issue?
  - demonstrating 'true' interoperability

What is SWORD doing next …

- agree a protocol, develop profile
  - 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

www.ukoln.ac.uk/repositories/digirep/index/SWORD
j.allinson@ukoln.ac.uk