Intrallect intraLibrary 3.1: Integration Guide

Revision: 6
Created: 1st August 2005
Last Revised: 30th June 2009
Contact: support@intrallect.com
Company: Intrallect Ltd
Product: intraLibrary, Digital Object Repository
Copyright: Copyright Intrallect Ltd 2003-2009. All rights reserved.

Table Of Contents

1. Introduction

This guide is for those responsible for managing software systems which will need to integrate with intraLibrary.

The following integration points are described in this guide.

Metadata Harvesting
: The metadata harvesting interface, based on the Open Archive Initiative - Protocol for Metadata Harvesting (OAI-PMH), allows the metadata from intraLibrary to be harvested in XML form for use in other systems such as catalogues, aggregators and archives.

Search and Retrieve
: The Search and Retrieve URL (SRU) and Search and Retrieve Web-Service (SRW) are related specifications, which both enable external systems to search intraLibrary and retrieve the search results.

File Transfer
: How organisations can automate the upload of packages and files.

Digital Repositories Interoperability
: This interface offers a subset of the repository services defined in the IMS Digital Repositories Interoperability Specification.

Repository Java-API
: Deeper access to the underlying functionality of intraLibrary, used for writing custom integration tools and extensions.

Authentication
: Various ways intraLibrary can share and exchange user authentication information with other systems.

2. Metadata Harvesting

An IntraLibrary repository can expose metadata for harvesting by other systems, using the Open Archive Initiative - Protocol for Metadata Harvesting (OAI-PMH). This section describes how intraLibrary supports OAI-PMH. For full details of the protocol visit the OAI web site. IntraLibrary supports version 2.0 of OAI-PMH.

2.1. Opening Collections for Harvesting

Any intraLibrary repository may contain multiple collections of digital objects. The collections in an intraLibrary repository are closed by default, which means records contained in the collection are not harvestable. An administrator-class user of intraLibrary can open the collection for harvesting by changing a single property of the collection. Individual collections are exposed as "sets" under OAI-PMH.

2.2. Entry Point

The entry point for harvesting metadata under OAI-PMH from intraLibrary is the root URL for the intraLibrary installation, with "/IntraLibrary-OAI" appended to it:

http://<intralibrary-root>/IntraLibrary-OAI

2.3. Metadata formats

The metadata formats that can be harvested from intraLibrary are summarised in the following table.

Name	Prefix	Specification
Dublin Core	dc	http://dublincore.org/documents/dces
IMS Metadata 1.2	imsmd	http://www.imsglobal.org/metadata/index.html
IEEE LOM	lom	http://ltsc.ieee.org/wg12/materials.html
ODRL (version 1.1)	o-ex	http://odrl.net/

2.4. Testing

You can test the harvestability of a repository using the OAI Registry Explorer. This allows you to send commands to intraLibrary. Type in the URL including */IntraLibrary-OAI* and then you can select commands such as "Identify", "List Metadata Formats", etc. Note that for some commands you need to use parameters.

2.5. Modifying the Metadata Returned

IntraLibrary can be configured to overwrite parts of the metadata harvested under OAI-PMH. In the "General Configuration" section of the "system" page there is an option to select a metadata template to apply to harvested metadata. The values and macros in the selected template will overwrite the values stored in the relevant fields of each record returned in a harvest operation.

Possible uses for the feature include:

- consistently exposing useful properties, such as PublicURLs in specific metadata fields
- blanking the contents of certain fields for privacy or security reasons
- ensuring consistency in important statements, such as those describing copyright or usage policy

3. Search and Retrieve Service

IntraLibrary supports two related specifications for search and retrieve services, both open specifications published by the Library of Congress:

- Search and Retrieve URL (SRU)
- Search and Retrieve Web-Service (SRW), also known as "SRU Web Services"

Both these specifications support searching a remote catalogue and retrieving search results. They use the same query language, Common Query Language (CQL) and similar XML formats for the results of an "explain" and "query" operation.

3.1. SRU

Search and Retrieve URL (SRU) is a REST-like interface. It can be called with a simple HTTP GET request, with parameters specified in the query string of the URL, and responds with an XML document. The entry point for the SRU service is:

- http://<intraLibrary-root>/IntraLibrary-SRU

3.2. SRW

The entry point for the SRW service is:

- http://<intralibrary-root>/services/SRW

The SOAP binding has been implemented as follows, based on guidance from the Library of Congress team and the standard WSDL for SRW:

- Service Style: document/literal
- Message Style: inline with no multi-refs
- SOAP version: 1.1

3.3. Queries

CQL

Queries in SRU and SRW are defined in Common Query Language (CQL). It is very straightforward to create simple queries in CQL, but it is also possible to create complex queries, with multiple constraints. IntraLibrary supports "Level 1" of CQL.

Searches can be constrained by specific metadata fields using search indices in the Dublin Core context set. There is currently no standard context set for IEEE LOM metadata. Support for some additional search indices has been added, which are described in the next section.

Some simple queries:

- fish
- dc.title=metadata
- dc.creator="john smith"
- dc.keyword=snow
- dc.subject="Natural history of organisms"

3.4. Additional Search Indices

IntraLibrary supports selected indices from additional CQL "context sets". It is anticipated that support for more context sets will be added in future versions of intraLibrary.

Record Metadata Context Set

The Record Metadata Context Set includes a range of "indices" covering various information that might recorded in the metadata of a digital object. The indices supported by intraLibrary are listed in the following table.

index name	description
rec.collectionName	Name of a collection which contains the record. This is set by editing the name of a collection in the "admin tools" interface of intraLibrary.
rec.collectionIdentifier	Identifier for a collection which contains the record. This is a new property of a collection, which can be set by editing the properties of a collection in the "admin tools" interface of intraLibrary.

The index *rec.collectionName* and *rec.collectionIdentifier* can be used as part of a query to specify particular collections. For example a query can:

- constrain a results set to objects in one collection only
  - dc.title = "cat" and rec.collectionName = "e-Prints"
- return the entire contents of a collection
  - rec.collectionIdentifier = "uk.ed.epr1"
- obtain results from more than one collection
  - dc.title = "cat" and (rec.collectionName = "images" or rec.collectionName = "learning objects")

NB: The *rec.collectionName* and *rec.collectionIdentifier* indices replace the "collection" context set which was previously made available for use with intraLibrary 2.8, and is now no longer available.

Examples of use

1. Not constrained by collection:
   - A basic search query for the word "yacht" might have the following URL:
     - http://myorg.intralibrary.com/IntraLibrary-SRU?query=yacht&startRecord=1&maximumRecords=10&recordSchema=dc&version=1.1&operation=searchRetrieve
2. Constraining by collection ID:
   - With added constraint of a collection of ID of "abc123", value of query parameter would become:
     - query=yacht AND rec.collectionIdentifier=abc123
   - URI encoded this would be:
     - query=yacht+AND+rec.collectionIdentifier%3Dabc123

- giving complete URL of:
  - http://myorg.intralibrary.com/IntraLibrary-SRU?query=yacht+AND+rec.collectionIdentifier%3Dabc123&startRecord=1&maximumRecords=10&recordSchema=dc&version=1.1&operation=searchRetrieve

3. Contraining by collection name

- With added constraint of collection name of "Sailing Pictures", value of "query" parameter would become:
  - query=yacht AND rec.collectionName="Sailing Pictures"
- URI encoded this would be:
  - query=yacht+AND+rec.collectionName%3D%22Sailing+Pictures%22
- giving:
  - http://myorg.intralibrary.com/IntraLibrary-SRU?query=yacht+AND+rec.collectionName%3D%22Sailing+Pictures%22&startRecord=1&maximumRecords=10&recordSchema=dc&version=1.1&operation=searchRetrieve

3.5. Record Extensions

IntraLibrary supports a range of extensions to record metadata returned by a SRU/SRW request. These are needed to support various additional functionality requested by various organisations using intraLibrary.

Where possible Intrallect has adopted existing extensions, such as parts of the Record Metadata Schema. Where Intrallect has added its own extensions, they have been assigned a unique namespace using Intrallect's own "authority string".

3.5.1. Record Metadata Schema

Record Metadata Schema

The Record Metadata Schema includes a range of extension elements covering various additional information that might be recorded in the metadata of a digital object. The extension elements supported by intraLibrary are listed in the following table.

element	description
modified	date/time when the record was last modified

created	date/time when the record was created

The following example shows how the "rec" extensions may appear in the context of a record returned as part of an SRU/SRW response. The "rec" prefix is mapped to the XML namespace "http://srw.o-r-g.org/schemas/rec/1.0/".

```
<record>
  <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
  <recordPacking>xml</recordPacking>
  <recordData>
    <srw_dc:dc>
       <dc:title>This is a Sample Record</dc:title>
    </srw_dc:dc>
  </recordData>
  <recordPosition>1</recordPosition>
  <extraRecordData>
     <rec:record>
        <rec:history>
           <rec:modification>
              <rec:modified>2006-12-09T12:10:00</modified>
              <rec:created>2004-11-09T13:11:21</created>
           </rec:modification>
        </rec:history>
      </rec:record>
  </extraRecordData>
</record>
```

3.5.2. Review Extensions

Intrallect has defined a set of elements to expose statistical information about the reviews/comments that have been made on objects in the repository. The "review" extensions are designed to be included in the "extraRecordData" element of the SRU/SRW response, mapped to the XML namespace "info:srw/extension/13/review-v1.0".

Element Name	Description

meanStarRating	the mean of all the star ratings (0-5) of the digital resource made by users of the repository
numberOfStarRatings	the total number of star ratings made on the digital resource by users of the repository
numberOfComments	the total number of comments made on the digital resource by users of the repository

The following example shows how the "review" extensions may appear in the context of a record returned as part of an SRU/SRW response. The "review" prefix is mapped to the XML namespace "info:srw/extension/13/review-v1.0".

```
<record>
  <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
  <recordPacking>xml</recordPacking>
  <recordData>
    <srw_dc:dc>
      <dc:title>This is a Sample Record</dc:title>
    </srw_dc:dc>
  </recordData>
  <recordPosition>1</recordPosition>
  <extraRecordData>
     <review:meanStarRating>3.5</review:meanStarRating>
     <review:numberOfStarRatings>10</review:numberOfStarRatings>
     <review:numberOfComments>17</review:numberOfComments>
  </extraRecordData>
</record>
```

3.5.3. Package Extensions

Intrallect has defined a set of elements to expose information about the packaging of digital objects in the intraLibrary repository system. The "package" extensions are designed to be included in the "extraRecordData" element of the SRU/SRW response, mapped to the XML namespace "nfo:srw/extension/13/package-v1.0".

Element Name	Description
packageType	The type of content package that is stored. A sample

	vocabulary for this field is given below.
packageTypeVersion	The version of the content packaging specification the package conforms to
packageDownloadLocator	A locator, typically a URI, but could be another resolvable identifier of any kind, from which the package can be downloaded, with the appropriate authorisation
packageManifestLocator	A locator, typically a URI, but could be another resolvable identifier of any kind, at which the manifest file of the package can be accessed in the context of the package as a whole. This allows a structured package to be delivered by a remote system, such as a Learning Management System (LMS), whilst the content itself remains in the repository. There are various scenarios where this could be valuable, including: • display a structured IMS or SCORM package inside a VLE/LMS, whilst it remains in the repository • making a link to a particular page inside a content package, such as an internal navigation page or index • handing the responsibility for playing/previewing particular kinds of packages to appropriate services or engines, such as • an IMS QTI player • an IMS Learning Design engine • a SCORM player or IMS Simple Sequencing engine • an IMS Common Cartridge player
packagePreviewLocator	A locator, typically a URI, but could be another resolvable identifier of any kind, which resolves to the built in "public preview" function of intraLibrary. The behaviour of the public preview depends on the type of objects that is stored: • single file: the file is returned for display by the browser or appropriate browser plugin or helper application • multi-file package: a frameset with the browsable structure of the package in the left-hand frame, and the first "page" of the package in the right-hand frame • web (URL) reference object: forwards to the stored URL • record of a "physical" object: a page displaying the stored location of the physical object

The following table defines a sample vocabulary for the "packageType" element. The current version of intraLibrary supports a subset of these package types and terms, as indicated in the table.

Term	Usage	Supported
ims-common-cartridge	zip file structured according to the IMS Common Cartridge specification (when released)	No
ims-cp	zip file structured according to the IMS Content Packaging specification, in its generic form	Yes
mets	zip file structured according to the Metadata Encoding and Transmission Standard (METS)	No
mpeg-didl	a digital object made up of multiple digital items, described using the Digital Item Declaration Language (DIDL), part of MPEG-21	No
scorm	zip file structured according to the ADL SCORM specification (based on IMS Content Packaging). Possible versions include "1.2" and "2004".	Yes

The following example shows how the "package" extensions may appear in the context of a record returned as part of an SRU/SRW response. The "package" prefix is mapped to the XML namespace "info:srw/extension/13/package-v1.0".

```
<record>
  <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
  <recordPacking>xml</recordPacking>
  <recordData>
    <srw_dc:dc>
      <dc:title>This is a Sample Record</dc:title>
    </srw_dc:dc>
  </recordData>
  <recordPosition>1</recordPosition>
  <extraRecordData>
    <package:packageType>imscp</package:packageType>
    <package:packageTypeVersion>1.1.2</package:packageTypeVersion>
    <package:packagePreviewLocator>
      http://repository.mydomain.edu/open-preview?key=i175a12642t
    </package:packagePreviewLocator>
    <package:packageManifestLocator>
      http://repository.mydomain.edu/objects/121/files/imsmanifest.xml
      </package:packageManifestLocator>
      <package:packageDownloadLocator>
         http://repository.mydomain.edu/download-package?object-id=121
      </package:packageDownloadLocator>
   </extraRecordData>
</record>
```

3.6. Metadata Formats

The metadata format returned by the SRU/SRW service can be controlled by setting the value of the "recordSchema" parameter in the SRU/SRW request. The following table gives details of the record formats currently supported in the SRU/SRW response.

Name	value	specification
Dublin Core	dc	http://dublincore.org/documents/dces
IEEE-LOM	lom	http://ltsc.ieee.org/wg12/materials.html

3.7. Access Control

Possible Approaches

The ability of external systems to search an intraLibrary repository is configurable for each collection within the repository. Two approaches are currently supported:

1. selectively opening/closing collections for external search
   - this is controlled by editing the properties of a collection in the "admin tools" interface
   - a collection is either open to external searching or closed to external searching (default)
2. controlling access using SRU authentication tokens
   - tokens are assigned to a collection by editing the properties of a collection in the "admin tools" interface
   - tokens can supplied in the search request using the "x-info-2-auth1.0-authenticationToken" parameter
   - results will only be returned from collections to which the specified token has been assigned, or to which no tokens have been assigned

Example of Using the Authentication Token

- A basic search query for the word "yacht" might have the following URL:
  - http://myorg.intralibrary.com/IntraLibrary-SRU?query=yacht&startRecord=1 &maximumRecords=10&recordSchema=dc&version=1.1&operation=searchRetrieve
- To apply the authentication token "xyz789, you would append the following to the URL:
  - &x-info-2-auth1.0-authenticationToken=xyz789
- giving:
  - http://myorg.intralibrary.com/IntraLibrary-SRU?query=yacht&startRecord=1 &maximumRecords=10&recordSchema=dc&version=1.1&operation=searchRetrieve &x-info-2-auth1.0-authenticationToken=xyz789

3.8. Modifying the Metadata Returned

The system can be configured to overwrite parts of the metadata returned in SRU/SRW respones. In the "General Configuration" section of the "system" page there is an option to select a metadata template to apply to search results. The values and macros in the selected template will overwrite the values stored in the relevant fields of each record returned in a search request.

Possible uses for the feature include:

- consistently exposing useful properties, such as PublicURLs in specific metadata fields
- blanking the contents of certain fields for privacy or security resasons
- ensuring consistency in important statements, such as those describing copyright or usage policy

3.9. Conformance

IntraLibrary complies with the Base Profile Requirements for SRW. It conforms with "Level 1" of CQL. Range queries are not currently supported.

IntraLibrary supports the DC context set, and returns metadata in Simple Dublin Core (default) and IEEE LOM formats.

4. File Transfer

Sometimes it is not practial to use the "upload" facility to put files and packages into intraLibrary. For example, a file/package may be too big to upload in a reasonable time over a restricted network collection, or there may be a large set of files or records that need to be transferred from an existing system.

4.1. SWORD

The Simple Web-service for Ordering Repository Deposit (SWORD) is REST web service based on the Atom Publishing Protocol (AtomPub). SWORD was developed by the JISC-funded SWORD Project, to which Intrallect was a founding contributor.

AtomPub was originally designed for the remote authoriong and editing of blog entries. In contrast, SWORD is primarily concerned with the deposit of files or packages into a repository. SWORD uses the AtomPub mechanism for posting media resources to a Collection (section 9.6) to achieve this.

The following is brief summary of how you can use Intrallect's implementation of SWORD to deposit fields into intraLibrary from other systems, desktop programs, or migration tools.

4.1.1. Entry Point

The entry point for the SWORD service provided by intraLibrary is:

- http://<intraLibrary-root>/IntraLibrary-Deposit

4.1.2. Authentication

SWORD is an authenticated service. Any contributor user accessing intraLibrary through the SWORD interface will effectively be using their own “Work Area”, giving them access to deposit into the same groups and collections that are available from the upload and import functions of the intraLibrary “Work Area”.

IntraLibrary supports using HTTP-Basic authentication for the SWORD web-service. The following authentication information must be supplied for any request to this service:

- ***username***
- ***password*** : in the clear, or using SHA-1 encryption
- ***realm*** : will normally be "IntraLibraryRealm"

4.1.3. Service Document

Any client that needs to explore the structure of the users work area, will need to download the AtomPub service document. This is an XML document which sets out which groups and collections are available for the user for the purpose of deposit. The URI for the service document is:

- http://<intraLibrary-root>/IntraLibrary-Deposit/service

The following XML sample is an example of a service document returned by intraLibrary. It features these basic concepts:

- ***group*** : A group of users in intraLibrary.
  - The <*workspace*> element indicates a group
  - All the groups through-which a user has deposit access, will be listed using separate <*workspace*> elements.
- ***collection*** : A collection in intraLibrary.
  - The <*collection*> element indicates a group
  - Within each workspace (i.e. group), the collections for which the user has deposit access will be listed.
  - The "href" attribute of each collection contains the URI to be used for any deposit request that targets the relevant group and collection.
- ***metadata*** : Brief information about the groups and collections.
  - The element <*atom:title*> contains the title of the collection or group
  - The element <*dcterms:abstract*> contains the description of the collection.
  - The element <*accept*> contains a list of mime-types that can be deposited into the collection.
    - Normally any mime-type can be deposited, as indicated by <*accept*>*/*</*accept*>
  - The element <*sword:formatNamespace*> contains the XML namespace of the packaging format supported for ingest by the repository.
    - For intraLibrary the value of this element is normally *http://www.imsglobal.org/xsd/imscp_v1p1*, which means the repository supports IMS and SCORM content packages.
  - The element <*sword:treatment*> contains a description of how a package of content

will be handled when deposited into the repository.

```
<?xml version="1.0" encoding="UTF-8"?>
<service xmlns="http://purl.org/atom/app#"
    xmlns:atom="http://www.w3.org/2005/Atom"
    xmlns:sword="http://purl.org/net/sword/"
    xmlns:dcterms="http://purl.org/dc/terms/">
  <workspace>
    <atom:title>Default</atom:title>
    <collection href="http://repository.mydomain.edu/IntraLibrary-Deposit/import/group1/collection1">
      <atom:title>open</atom:title>
      <accept>*/*</accept>
      <dcterms:abstract>the default collection</dcterms:abstract>
      <sword:treatment>Zip archives recognised as content packages are opened and the
      individual files contained in them are stored.  All other files are stored as is.</sword:treatment>
      <sword:formatNamespace>http://www.imsglobal.org/xsd/imscp_v1p1</sword:formatNamespace
    </collection>
  </workspace>
  <workspace>
    <atom:title>sword</atom:title>
    <collection href="http://repository.mydomain.edu/IntraLibrary-Deposit/import/group7/collection8">
      <atom:title>sword</atom:title>
      <accept>*/*</accept>
      <dcterms:abstract>for sword testing</dcterms:abstract>
      <sword:treatment>Zip archives recognised as content packages are opened and the
      individual files contained in them are stored. All other files are stored as is.</sword:treatment>
      <sword:formatNamespace>http://www.imsglobal.org/xsd/imscp_v1p1</sword:formatNamespace
    </collection>
    <collection href="http://repository.mydomain.edu/IntraLibrary-Deposit/import/group7/collection1">
      <atom:title>open</atom:title>
      <accept>*/*</accept>
      <dcterms:abstract>the default collection</dcterms:abstract>
      <sword:treatment>Zip archives recognised as content packages are opened and the
      individual files contained in them are stored. All other files are stored as is.</sword:treatment>
      <sword:formatNamespace>http://www.imsglobal.org/xsd/imscp_v1p1</sword:formatNamespace
    </collection>
  </workspace>
  <sword:verbose>true</sword:verbose>
  <sword:level>1</sword:level>
  <sword:noOp>true</sword:noOp>
</service>
```

4.1.4. Deposit Request

The deposit operation is a simple HTTP POST request:

- The *target URI* of the POST request should be the URI given in the href attribute of the relevant <collection> element in the service document.
  - The path of this URI contains all the information about the group and collection that should be used for the deposit.
- The body of the request should contain the resource that is to be deposited in the repository.
- HTTP-Basic authentication information must be supplied with the request.
- Normal HTTP headers, such as *Content-Type* and *Content-Length* should be supplied.

Any other parameters of the deposit operation are supplied in additional HTTP headers, including some extensions designed specifically for SWORD. The following optional parameters may be used in the HTTP request for a SWORD deposit:

- ***Content-Disposition*** : used to supply the name of the file being deposited
  - Necessary if you want the file retain its original name when downloaded or exported from the repository.
  - used in the following form:
    - Content-Disposition: filename=[name of file]
- ***Content-MD5*** : used to supply an MD5 checksum (hash) of the file.
  - When used, this enables the repository to check the file it receives has not been corrupted.
  - If an MD5 checksum is supplied in the HTTP request, and the checksum generated from the file received by the repository does not match the checksum supplied in the HTTP request, an error will be returned (HTTP response code 412).
- ***X-On-Behalf-Of*** : the username of the user on whom's behalf the resource is being deposited.
  - Used by external systems, such as a Learning Management System (LMS), depositing resources on behalf of its users (a.ka. mediated deposit). The external system must authenticate as an "administrator" class user to get access to this feature.
- ***X-Format-Namespace*** URI of the XML namespace of the packaging format used for the resource (if any).
  - Only necessary if you want the repository to reject packages which do not use this

namespace.

- ***X-Verbose*** : Possible values of "true" or "false" (default).
    - To return a more verbose report of the treatment of, or any problems with the deposit, in the <*sword:verboseDescription*> of the response
- ***X-No-Op*** : Possible values of "true" or "false" (default).
    - To test the behaviour of deposit operation, or a particular resource, without actually storing the resource in the respository.

4.1.5. Deposit Response

If a deposit operation has succeeded, then a HTTP response with a response code of 201 will be returned. In many situations, this could be the only information that is needed by the deposit client.

To provide more information, an Atom "entry" document is included in the body of the response. The following XML sample is an example of an Atom "entry" document returned by intraLibrary, in response to a successful deposit request. The following elements contain information that could be useful:

- ***<atom:title>*** : The title given to the resource in the repository.
    - If the resource is a content package, the title could have been retrieved from metadata stored in the package.
    - Otherwise, the title will normally be the filename of the resource, if supplied using the *Content-Disposition* header.
- ***<atom:id>*** : the OAI-style identifer generated for the resource, prefixed with "atom:".
- ***<atom:author>*** : the child element <*atom:name*> will contain the full name of the final owner of the resource.
- ***<atom:contributor>*** : the child element <*atom:name*> will contain the full name of the final owner of the resource.
- ***<atom:content>*** : the *src* attribute of this element is populated with the Public Preview URL of the deposited resource.
- ***<atom:generator>*** : contains the name of the repository, and indicates the base URI of the repository in the "uri" attribute.

If the relevant optional parameters have been used, the following elements could also appear:

- ***<sword:verboseDescription>*** : a more verbose report of the treatment of, or any problems with the deposit

In accordance with the SWORD Application Profile, the following elements are included to retain compliance with the AtomPub specification, but the implied functionality is not yet available.

- ***<atom:link rel="edit-media">*** :
- ***<atom:link rel="edit">*** :

```
<?xml version="1.0" encoding="UTF-8"?>
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:sword="http://purl.org/net/sword/">
  <atom:title>integration_guide.html</atom:title>
  <atom:id>atom:com.intrallect.beans.3p0:1452</atom:id>
  <atom:updated>2008-07-04T14:47:57Z</atom:updated>
  <atom:author>
    <atom:name>John Smith</atom:name>
  </atom:author>
  <atom:contributor>
    <atom:name>John Smith</atom:name>
  </atom:contributor>
  <atom:summary type="text" />
  <atom:content src="http://repository.mydomain.edu/IntraLibrary?command=open-preview&learning_
  <atom:link rel="edit-media" href="http://repository.mydomain.edu/IntraLibrary-Deposit/edit-media/lea
  <atom:link rel="edit" href="http://repository.mydomain.edu/IntraLibrary-Deposit/edit/learning_object_
  <atom:source>
    <atom:generator uri="http://repository.mydomain.edu">Intrallect Test (pt3)</atom:generator>
  </atom:source>
  <sword:verboseDescription>File:integration_guide.html was imported</sword:verboseDescription>
</atom:entry>
```

If the a deposit operation has failed, then a variety of response codes could be returned, depending on the reason for the failure. As far as possible, normal HTTP response codes are used for their standard purposes. Details of the use of HTTP response codes used in SWORD are described in section 5.5 of the SWORD Application Profile.

4.1.6. Conformance

The SWORD specification has two levels of compliance, Level 0 and Level 1. Level 0 of SWORD is a simple subset of the AtomPub. Level 1 of SWORD adds some simple extensions to the HTTP requests and response documents defined in Level 0. There are some other optional features to support developers and implementors. IntraLibrary fully supports Level 1 of SWORD, including all the optional features, so that is what is documented here.

4.2. Alternatives

It is possible to provide other means for users to transfer files to the repository server, such as FTP, including Secure FTP (SFTP) and WebDav. The "file import" action in an intraLibrary workflow allows users to import files from a specified folder on the server filesystem. The location of this folder is configurable by user group. Any FTP or WebDAV server can be used to give intraLibrary users controlled access to allow them to transfer files to the appropriate folder on the server. If you would like some more information about this, please contact Intrallect.

It is possible to automate the deposit of files/packages into the repository using the Repository Java API. A sample implementation is provided with the product.

5. IMS DRI

The implementation of the IMS Digital Repository Interoperability (IMS DRI) "spec" in intraLibrary is based on a specification developed by the (ECL) project. This interface is deprecated, but still maintained for backward compatibility.

6. Repository Java-API

IntraLibrary has a fully documented "repository" Java-API. This API enables Java classes running in the same application context (webapp) as intraLibrary to access aspects of the intraLibrary business layer. It does not support Remote Method Invocation (RMI).

The Repository Java-API can be used to enable other software to:

- gather information about the status of a repository and its contents
    - All the key data objects are now exposed through the Java API, facilitating the generation of custom reports
- create and manage users
- trigger workflow events

- query the structure of a workflow and find learning objects within it
- search for objects

The following support information is provided with the product:
- complete set of JavaDocs
- some sample implementations, including full source code and installation instructions:
    - adding users to intraLibrary
    - uploading multiple files into intraLibrary
    - searching for learning objects in intraLibrary

7. Authentication

7.1. External Authentication

In its default configuration IntraLibrary uses a built-in directory of user accounts when authenticating users. IntraLibrary can also be configured to query a chain of external sources of authentication when authenticating users. This means an intraLibrary repository may be used by an organisation that has one or more existing directories of users, or by a group of organisations working together, who each want to manage their user accounts seperately.

An authentication chain is defined in a simple XML file in the configuration folder of the intraLibrary installation. When a user attempts to log-in to intraLibrary, their credentials are checked against every source of authentication in turn. A chain will typically include the internal directory of users in intraLibrary itself.

There are many possible kinds of external sources of authentication, including:
- a set of user accounts managed via the operating system of a local fileserver
- a database of users that is part of another enterprise application
- a directory exposed under the Lightweight Directory Access Protocol (LDAP) or Microsoft Active Directory
- a regional or national authentication service
- a federated authorisation mechanism, such as Shibboleth

IntraLibrary supports a pluggable authentication interface. An implementation of this authentication interface is required for each kind of authenticator used in an authentication chain. Full documenation of this inteface is provided with the system, including the following reference implementations:

- a simple SQL-based authenticator
- an LDAP authenticator
- an integration with EduServ Athens Single-Sign On

7.2. Re-Using IntraLibrary Authentication

The intraLibrary authentication solution can be re-used by other services and applications running alongside intraLibrary. Specifically, it is made available as an authentication "realm" within Apache Tomcat.

The intraLibrary Realm can be applied within a "Engine", "Host" or "Context" element of the Tomcat configuration, taking account of the rules on scope, using the following syntax:

```
<Realm className="com.intrallect.realm.IntraLibraryRealm"
      configDir="[path of intraLibrary configuration directory]"/>
```

8. Using Events

IntraLibrary now exposes an "Event Handling" framework in which customized actions can be taken by customer-defined “event handlers”. When a users takes an action in the repository system, such as doing a search, this triggers the relevant "event". A custom event-handler can retrieve information about all the objects related to that event, and trigger other actions, such as logging or notification.

Event handlers are written in Java, and must implement the "IntraLibraryEventHandler" interface. Full documentation and examples are provided with the system, including a comprehensive list of all the event types and the information that is available about them. Event handlers are registered with IntraLibrary using an xml descriptor file. IntraLibrary ensures that each eventHandler is notified when an event occurs.

9. Future Developments

Intrallect plans to implement the following interfaces in intraLibrary:

- Implementation of OpenAthens SP for protocol independent authentication through Athens.
- Harvesting from other repositories via OAI-PMH.
- Support the "scan" verb of SRU, to expose vocabularies to other systems
- Searching other catalgoues via SRU/SRW.
- Enable an intraLibrary repository to be searched via the FIRE interface.
- Update the SRU interface from SRU 1.1 to SRU 1.2

Intrallect is considering the following interfaces for implementation in intraLibrary:

- Federated authorisation using Shibboleth
- Transfer content packages to/from intraLibrary using AICC's PENS specification
- Manage user accounts in intraLibrary using the IMS Enterprise specification
- Open URL

10. References

- Open Archive Initiative - Protocol for Metadata Harvesting (OAI-PMH): http://www.openarchives.org/pmh/
- OAI Repository Explorer: http://re.cs.uct.ac.za/
- Search and Retrieve URL (SRU): http://www.loc.gov/standards/sru/sru-spec.html
- Search and Retrieve Web-Service (SRW): http://www.loc.gov/standards/sru/srw
- Dublin Core Metadata: http://dublincore.org/documents/dces
- IMS Metadata: http://www.imsglobal.org/metadata/index.html
- IEEE LOM: http://ltsc.ieee.org/wg12/materials.html
- Open Digital Rights Language (ODRL): http://odrl.net/
- Simple Web-service for Ordering Repository Deposit (SWORD): http://purl.org/net/sword/
- Atom Publishing Protocol (AtomPub): http://bitworking.org/projects/atom/rfc5023.html
- IMS Digital Repositories Interoperability Specification: http://www.imsglobal.org/digitalrepositories