Use of Repository Deposit Protocol

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Table of Contents

Introduction

Institutional Repositories are increasingly used as archives for publications, theses and other scholarly or educational output. With this, the archival services offered by repository applications maintained by the university libraries is used in an increasing number of business processes, whether this being the deposit of peer reviewed stage 2 material, the deposit of bachelor theses at the and of the academic year or the need to archive research data.
With this, library staff and repository applications need to accommodate more and more use cases, each requiring its specific needs, e.g. the deposit of a thesis resulting from a collaborative effort for publication in the HBO Kennisbank or the self-archival of research data at an external data repository.
In order to create a scalable solution, the need arises to separate the archival service from services that support the deposit workflows and to loosely couple institutional repositories with specialised deposit application which then exchange information through standardised web service interfaces.
This application profile describes a protocol to allow the deposit of publications, theses and research data to institutional repositories.

Standards reference for this application profile

SWORD: Simple Web service Offering Repository Deposit

The first version of Simple Web service Offering Repository Deposit was developed in a project funded by JISC as an effort to 'lowering the barriers to deposit, principally for depositing content (any content!) into repositories, but potentially for depositing into any system which wants to receive content from remote sources'. It is in itself an application profile of the Atom Publishing Protocol (AtomPub) described in RFC 5023 used for the publishing and editing of web resources using HTTP and XML 1.0. Although related, the AtomPub protocol is not to be confused with the Atom Syndication Format described RFC 4287 commonly used for web feeds used to check for updates on a website.

The current version of SWORD at the time of writing is version 1.3, although a new 2.0 version is in the making. This version however still has a working beta status and has not been finalised.

This application profile is based on #SWORDv1.3.

Part A: Use of SWORD features

This section will describe the use of the SWORD profile in the context of the SURFshare programme. The contents are organised according and supplementary to the document #SWORDv1.3 Part A. If a SWORD profile section or feature is omitted, implementations MUST behave as defined in SWORD profile.

1. Package Support

1.1. Package support in Service Description

The server MAY support multiple packaging formats with varying quality values according to the support of the Common Submission Information Package (SIP) specified in #Appendix 1.

The server MUST support at least one package format with Quality Value "1.0", indicating full support where all components supplied within the SIP will be processed and understood when using the designated package format.

All supported formats MUST be listed in the Service Document.

All formats listed in the Service Document MUST have a Quality Value attribute assigned.

The value used in the `<sword:acceptedPackaging>` element MUST NOT overload any values enumerated in the SWORD Content Package Types.

The server MAY use the `<sword:service>` element in the Service Document to indicate the existence of other service interfaces supporting additional package formats.

The server SHOULD NOT accept a specific package format across multiple interfaces with different levels of support as indicated by the Quality Value attribute in the Service Document.

1.2 Package support during Resource Creation

When describing packaged resources in Media Entry documents, the server SHOULD add sword:packaging elements to the entry.

1.3 Package description in entry documents

If a server receives a POST request with a format that is not listed as an accepted format in the Service Document, the server MUST reject the package by returning an #HTTP 415 (unsupported media type).

2 Mediated Deposit

⚠ todo

2.1 Mediation in Service Description

Servers supporting mediated deposit MUST indicate this by including a `sword:mediation` element with a value of "`true`" in the Service Document as defined in #SWORDv1.3 section 2.1.
For servers that do not include a SWORD mediation element in the Service Document, a default value of "no" SHOULD be assumed by clients.

4 Auto Discovery

AtomPub makes no recommendations on the discovery of Service Documents.
The SWORD profile states that it is RECOMMENDED that server implementations use an `<html:link rel="sword" href="http://example.org/sword/servicedocument.atom"/>` element in the head of a relevant HTML document to assist with service discovery.
In addition, it is RECOMMENDED to also include an `<atom:link rel="sword" type="application/atomsvc+xml" href="http://example.org/sword/servicedocument.atom"/>` element in relevant response documents such as Error Document.

6 Nested Service Description

Nested Service Descriptions MAY be used to specify alternative collections for both organisational (i.e. generic collection with a nested discipline specific collection) and technical purposes (i.e. a specific interface or service instance to cater for specific types of content packaging).

Part B: Use of AtomPub features

The contents are organised according and supplementary to the document #SWORDv1.3 Part B, which is turn is organised according to the sections of RFC 5023 describing #AtomPub. If a #SWORDv1.3 section or feature is omitted, implementations MUST behave as defined in #SWORDv1.3 profile.

9. Creating and Editing Resources

When depositing resources using SWORD, resources are created by a server when a client makes an HTTP POST request with the resource in the HTTP request body. If the deposit is made successfully, the server then gives a HTTP reponse with the #HTTP 201 status code in the header of the response indicating the resource has been successfully created at the repository side.

Servers returning a #HTTP 201 status code after a deposit MUST preserve the resource deposited.

Clients receiving a #HTTP 201 status code MUST consider the resource deposited as being accepted for storage by the repository.

9.2.2. Asynchronous treatment of resources

It MAY however be the case that the repository implements an additional asynchronous validation process after which a resource MAY or MAY NOT be accepted. This for instance is the case when a repository uses an intermediate repository where resources deposited through the SWORD interface are temporarily stored, after which they will be moved to a final location within the repository when they are properly validated by a repository manager. When a resource is then being rejected by the repository during the validation process after the server has sent an #HTTP 201 response to the client, the situation MAY arise where the client considers the resource as being successfully deposited into the repository, while in fact the resource is NOT being stored into the repository. This situation is viewed as undesirable.

Servers implementing an asynchronous validation process MUST return an #HTTP 202 Accept response code indicating the request has been accepted for processing, but the processing has not been completed.

Clients receiving a #HTTP 202 status code upon deposit of a resource MUST consider the resource deposited as NOT being stored into the repository.
RFC2616 states that there is no facility for the re-sending of status codes. Therefore, a client will not receive a notification of the outcome of the processing carried out by the server. In order to allow clients to retrieve the outcome of the deposit, the sword:treatment element MAY contain the status of the processing of the deposited resource.

Servers implementing #HTTP 202 status codes MUST supply a permanent link to the Atom Entry document of the response.

Servers implementing #HTTP 202 status codes MUST update the sword:treatment element of the Atom Entry document of the resource with the status of the processing of the deposited resource.

Client SHOULD implement a mechanism to confirm the successful deposit by periodically checking back at the server with an HTTP GET request to the permanent link supplied by the server, in order to check the contents of the `sword:treatment` element of the Atom Entry describing the deposited resource when a #HTTP 202 status code has been received upon deposit.

14. Securing the Atom Publishing Protocol

The SWORD profile states servers SHOULD support the use of HTTP Basic Authentication over TLS. In many institutional security policies, using basic authentication over unencrypted connections. de facto sending authentication credentials in plain view is viewed insufficient. Therefore this requirement has been restated as follows:

Servers implementing SWORD MUST support HTTP Basic Authentication (RFC 2617) over TLS (RFC 2818).

Appendix 1: Packaging

Appendix 2: Examples

References

SWORDv1.3. Allinson, J et al, "SWORD AtomPub Profile version 1.3". http://www.swordapp.org/docs/sword-profile-1.3.html

AtomPub. Gregario, J. and B. de hOra, "The Atom Publishing Protocol", RFC 5023, October 2007. http://www.ietf.org/rfc/rfc5023.txt (see also non-normative html version at http://bitworking.org/projects/atom/rfc5023.html)

HTTP1.1 Fielding et al, "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616, June 1999 http://www.w3.org/Protocols/rfc2616/rfc2616.html

- HTTP 201. Created. http://tools.ietf.org/html/rfc2616#section-10.2.2
- HTTP 202. Accepted. http://tools.ietf.org/html/rfc2616#section-10.2.3
- HTTP 415. Unsupported Media Type. http://tools.ietf.org/html/rfc2616#section-10.4.16