SWORD 2.0 Profile

Credits

SWORD 2.0 Technical Lead: Richard Jones, Cottage Labs

SWORD 2.0 Community Manager: Stuart Lewis, University of Auckland

SWORD 2.0 Technical Advisory Group
Julie Allinson, University of York
Tim Brody, University of Southampton
David Flanders, JISC
Graham Klyne, University of Oxford
Alister Miles, University of Oxford
Ben O'Steen, Cottage Labs
Mark MacGillivray, Cottage Labs
Rob Sanderson, LANL
Nick Sheppard, Leeds Metropolitan University
Eddie Shin, MediaShelf
Ian Stuart, University of Edinburgh
Ed Summers, Library of Congress
David Tarrant, University of Southampton
Graham Triggs, BioMed Central
Scott Wilson, University of Bolton

Further acknowledgements of input
Aaron Birkland (Cornell University), Tim Donohue (DuraSpace), Jim Downing (University of Cambridge), Ross Gardler (OSS Watch), Steve Midgley (US Department of Education), Glen Robson (National Library of Wales), Peter Sefton (University Of Southern Queensland), Adrian Stevenson (UKOLN), Paul Walk (UKOLN), Nigel Ward (University of Queensland)

1. Introduction

The Atom Publishing Protocol ([AtomPub]) provides a simple, extensible mechanism for the creation of Web Resources. The first major version of SWORD [SWORD 1.3] built upon the Resouce creation aspects of AtomPub to enable package deposit onto a server.

This approach of fire-and-forget deposit, where the depositor has no further interaction with the server is of significant value in certain use cases, but there are others where this is insufficient. Consider, for example, that the depositor wishes to construct a digital artifact file by file over a period of time before deciding that it is time to archive it. In these cases, a higher level of interactivity between the participating systems is required, and this is the role that SWORD 2.0 is intended to fulfil.

The SWORD 2.0 profile continues to build upon AtomPub and HTTP ([RFC2616]) through the inclusion of the Create,

Retrieve, Update and Delete (CRUD) operations of AtomPub, in order to enable the following kinds of interactions:

- Clients may create resources by sending compound resources, such as archive files (tar, zip).
- Both non-interactive and 3rd party mediated operation are required.
- Workflows which may or may not include manual stages before deposited resources become available as web resources, are acknowledged and supported
- Clients may update or delete the compound resources or associated metadata

The role that SWORD aims to fulfil is that of a thin integration layer between any two scholarly systems aiming to exchange content, but not to provide tight integration between them. For standards to integrate detailed content management operations it is recommended to look at OASIS CMIS [CMIS], or the GData API [GData]. In contrast to these, SWORD is a deposit lifecycle standard, which supports the users and the participating systems in effectively exchanging content. Most of the enhancements in SWORD 2.0 over SWORD 1.3 are around closing the deposit loop to give the client software the opportunity to provider a richer environment to the user.

The scope of SWORD v2 will be limited to the deposit process between any two scholarly systems or between a user facing system and a service provider. This deposit process is only a portion of the full content lifecycle and does not attempt to provide support for collaborative or distributed authoring environments or policy management; it is focused entirely on the process of moving content from one location to another.

The deposit lifecycle encompasses the process of delivering content to a SWORD server which may have an ingest workflow in place. The content that has been deposited is then considered to be in the deposit process for the duration of that ingest workflow, during which time the client system may wish to interact with the submission. It may be that the client needs to make changes to the submission during the workflow process, either autonomously or when prompted for additional input by the server. Or it may be that the client wants to track the progress of the submission, at its leisure. Eventually the content will be formally accepted into the system and the client needs a way to know that this is happened and a route to follow to locate the deposited content.

The practical details of the extensions used in the SWORD Profile are provided in the following Internet Drafts:

1. Packaged Content Delivery over HTTP [SWORD001]
2. AtomPub Extensions for Packaged Content [SWORD002]
3. AtomPub Extensions for Scholarly Systems [SWORD003]
4. Atom Multipart Extensions for Packaged Content [SWORD004]

2. Notational Conventions

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 [RFC2119].

3. Terminology

Throughout this document we will regularly refer to identifiers which are used by SWORD. The following IRIs are part of the specification:

SD-IRI
: The **Service Document IRI**. This is the IRI from which the root service document can be located. If the server supports nesting of service documents, then these sub-service documents are also subject to the same rules as **SD-IRI** and should be considered the same.

Col-IRI
: The IRI of a **SWORD Collection**. This is the IRI to which the initial deposit will take place, and which are listed in the **Service Document**.

Cont-IRI
: The **Content IRI**. This is the IRI from which the client will be able to retrieve representations of the object as it resides in the SWORD server (which representation is returned will be through package format negotiation, as discussed later).

EM-IRI
: The **Atom Edit Media IRI**. This is almost always going to be the same as the **Cont-IRI**, but the Atom spec requires that these identifiers be allowed to differ [AtomPub]. Throughout this document we will mostly refer to the **EM-IRI**. An important thing to note about the **EM-IRI** and the **Cont-IRI** is that they do not refer to the resource as deposited by the client (e.g. the original zip), but to all the content stored in the item on the server, and the format that this IRI returns is obtained by content negotiation. This will be discussed in more detail in Section 10. Deposit Receipt.

Edit-IRI
: The **Atom Entry Edit IRI**. This is the IRI of the Atom Entry of the object, and therefore also of the container within the SWORD server.

SE-IRI
: The **SWORD Edit IRI**. This is the IRI to which clients may POST additional content to an Atom Entry Resource. This MAY be the same as the **Edit-IRI**, but is defined separately as it supports HTTP POST explicitly while the **Edit-IRI** is defined by [AtomPub] as limited to GET, PUT and DELETE operations.

State-IRI
: The **SWORD Statement IRI**. This is one of the IRIs which can be used to retrieve a description of the object from the sword server, including the structure of the object and its state. This may be the same as the IRI used for content management operations that the server implements beyond the behaviours defined by SWORD. See Setion 11: The SWORD Statement for more details.

Note that although there are 7 IRIs defined above, only 2 of them are added by SWORD: **State-IRI** and **SE-IRI**, the rest are replicated from [AtomPub] and are labelled here for convenience. Furthermore, the **SE-IRI** MAY be the same as the

Edit-IRI. Meanwhile the **EM-IRI** may be the same as the **Cont-IRI**, so implementers can implement the entire specification with only 5 differentiable IRIs.

4. Namespaces

4.1. SWORD Namespaces

All SWORD extensions are defined within the namespace:

```
http://purl.org/net/sword/
```

Beneath this URI, we use the following extensions for the various parts of the standard:

For all predicates associated with SWORD:

```
http://purl.org/net/sword/terms/
```

This document uses the prefix **sword** for this namespace name; for example **sword:originalDeposit**

Root registry for any SWORD recognised package formats including "zip" and "binary":

```
http://purl.org/net/sword/package/
```

Root namespace for any Error documents which SWORD can produce:

```
http://purl.org/net/sword/error/
```

Root namespace for any State terms that SWORD wishes to provide:

```
http://purl.org/net/sword/state/
```

4.2. Referenced Namespaces

The AtomPub [AtomPub] namespace is:

```
http://www.w3.org/2007/app
```

This document uses the prefix **app** for the namespace name; for example **app:collection**.

The Atom Syndication Format [Atom] namespace is:

```
http://www.w3.org/2005/Atom
```

This document uses the prefix **atom** for the namespace name; for example **atom:feed**

The DCMI Terms [DublinCore] namespace is:

```
http://purl.org/dc/terms/
```

This document uses the prefix **dcterms** for the namespace name; for example **dcterms:title**

The RDF [RDF] namespace is:

```
http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

This document uses the prefix **rdf** for the namespace name; for example **rdf:Description**

The OAI-ORE [OAIORE] namespace is:

```
http://www.openarchives.org/ore/terms/
```

This document uses the prefix **ore** for the namespace name; for example **ore:aggregates**

5. IRIs

The packaging format which represents a binary file which should not be unpacked by the server

```
http://purl.org/net/sword/package/Binary
```

The primitive SWORD packaging format, which represents a plain ZIP file during resource creation or update, and in Atom Multipart [AtomMultipart] deposit indicates that the Media Part is a plain ZIP file.

```
http://purl.org/net/sword/package/SimpleZip
```

6. Protocol Operations

While specific HTTP status codes are used below, a SWORD client should be prepared to handle any status code as per the HTTP status code definitions [RFC2616]

6.1. Retrieving a Service Document

See Sections 5.1 and 8 of [AtomPub], for details.

1. The client sends a GET request to the IRI of the Service Document.
2. The server responds with a Service Document enumerating the IRIs of a group of Collections and the capabilities of those Collections supported by the server. The content of this document can vary based on aspects of the client request, including, but not limited to, authentication credentials.

The client may also supply the `On-Behalf-Of` header [SWORD001] to provide the username of a target user on whose behalf a deposit will be made. See Section 8: Authentication and Mediated Deposit for more details.

```
GET SD-IRI HTTP/1.1
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Host: example.org
On-Behalf-Of: [username]
```

When a server that supports mediated deposit receives an `On-Behalf-Of` header [SWORD001], the returned Service Document SHOULD identify only those collections to which the combination of mediated user and authenticated user might successfully deposit. If the target server does not support mediated deposit, the returned Service Document SHOULD contain a `sword:mediation` [SWORD002] extension element with a value of `false` .

The service document supplied in response MUST meet the following criteria in addition to those specified in [AtomPub]:

- The SWORD server MUST specify the `sword:version` element with a value of `2.0` [SWORD003] as a child of the `app:service` element.
- The SWORD server MAY specify the `sword:maxUploadSize` (in kB) of content that can be uploaded in one request [SWORD003] as a child of the `app:service` element. If provided this MUST contain an integer.
- The SWORD server MUST specify the `app:accept` element for the `app:collection` element. If the Collection can take any format content type, it should specify `*/*` as its value [AtomPub]. It MUST also specify an `app:accept` element with an `alternate` attribute set to `multipart-related` as required by [AtomMultipart]. The formats specified by `app:accept` and `app:accept@alternate="multipart-related"` are RECOMMENDED to be the same.
- The SWORD server MAY include one `sword:collectionPolicy` [SWORD003] as a child of the `app:collection` element
- The SWORD server SHOULD include one `sword:mediation` element as a child of the `app:collection` element with a value of `true` or `false` [SWORD002]
- The SWORD server MAY include one `sword:treatment` element [SWORD003]as a child of the `app:collection` element
- The SWORD server MAY include zero or more `sword:acceptPackaging` elements [SWORD002] as a children of the `app:collection` element. The value SHOULD be a IRI for a known packaging format (where such IRIs exist)
- The SWORD server MAY include zero or more `sword:service` [SWORD003] elements as children of the `app:collection` element containing references to nested service descriptions

Example:

```
HTTP 1.1  200 OK
Content-Type: application/atomserv+xml

<?xml version="1.0" ?>
<service xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:sword="http://purl.org/net/sword/terms/"
    xmlns:atom="http://www.w3.org/2005/Atom"
    xmlns="http://www.w3.org/2007/app">

    <sword:version>2.0</sword:version>
    <sword:maxUploadSize>16777216</sword:maxUploadSize>

    <workspace>
        <atom:title>Main Site</atom:title>

        <collection href="http://swordapp.org/col-iri/43">
            <atom:title>Collection 43</atom:title>
            <accept>*/*</accept>
            <accept alternate="multipart-related">*/*</accept>
            <sword:collectionPolicy>Collection Policy</sword:collectionPolicy>
            <dcterms:abstract>Collection Description</dcterms:abstract>
            <sword:mediation>false</sword:mediation>
            <sword:treatment>Treatment description</sword:treatment>
            <sword:acceptPackaging>http://purl.org/net/sword/package/SimpleZip</sword:acceptPackaging>
            <sword:acceptPackaging>http://purl.org/net/sword/package/METSDSpaceSIP</sword:acceptPackaging>
            <sword:service>http://swordapp.org/sd-iri/e4</sword:service>
        </collection>
    </workspace>
</service>
```

The following requirements are placed on the client in receipt of a service document:

- If no `sword:maxUploadSize` is specified, the client MUST assume that no size limit exists, but this specification RECOMMENDS that clients limit themselves to reasonably small file sizes unless special arrangements with the server have been made.
- If no `sword:mediation` element is specified, the client MUST assume a value of `false` .
- If no `sword:acceptPackaging` element is specified, the client MUST assume that the server will not formally support packaging formats (see Section 5: IRIs and Section 7: Packaging for more details).
- The client SHOULD NOT attempt to deposit files with a packaging format that is not in the `sword:acceptPackaging` elements, although the client MAY specify the binary package format (see Section 5: IRIs and Section 7: Packaging for more details) in order to deposit opaquely packaged content

6.2 Listing Collections

See Section 5.2 of [AtomPub] for details.

AtomPub states that Collection Resources MUST provide representations in the form of Atom Feed Documents. Under the SWORD profile, implementations SHOULD provide such representations. Clients MUST NOT require a Collection Feed Document for deposit operation.

6.3. Creating a Resource

6.3.1. Creating a Resource with a Binary File Deposit

See Sections 5.3 and 9.2 of [AtomPub] for details.

The client can create a resource by POSTing the binary content as the body of an HTTP request to the **Col-IRI**, with the `Content-Type` , `Content-Disposition` , and `Packaging` headers, with the following requirements:

- The client SHOULD supply a `Content-Type` header
- The client MUST supply a `Content-Disposition` header with a `filename` parameter (note that this requires the filename be expressed in ASCII).
- The client SHOULD supply a `Content-MD5` header with the MD5 checksum hex encoded for the binary content
- The client SHOULD supply a `Packaging` header [SWORD001] providing the IRI (or other allowed) of the packaging format used. See Section 7: Packaging for more details.
- The client MAY provide an `In-Progress` header with a value of `true` or `false` [SWORD001]. See Section 9: Continued Deposit for more details.
- The client MAY provide an `On-Behalf-Of` header [SWORD001]. See Section 8: Authentication and Mediated Deposit for more details.
- The client MAY supply a `Slug` header providing a suggested identifier for the deposited content

```
POST Col-IRI HTTP/1.1
Host: example.org
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Type: application/zip
Content-Length: [content length]
Content-MD5: [md5-digest]
Content-Disposition: attachment; filename=[filename]
Packaging: http://purl.org/net/sword/package/METSDSpaceSIP
In-Progress: true
On-Behalf-Of: jbloggs
Slug: [suggested identifier]

[request entity]
```

The requirements placed on the server here are:

- If no `Content-Type` header is supplied the server MUST behave as per Section 7.2.1 in [RFC2616]
- The server SHOULD verify the `Content-MD5` header against the content. If the check fails, the server MUST respond with 412 Precondition Failed. See Section 12: Error Documents.
- Server implementations MUST adopt the behaviour and requirements in [RFC2183] with respect to the `Content-Disposition` header.
- If there is no `Packaging` header, the server SHOULD assume that it is the binary packaging format (See Section 5: IRIs
- Server implementations MUST adopt the behaviour and requirements in Section 9.7 of [AtomPub] with respect to the `Slug` header.
- If there is no `In-Progress` header, the server MUST assume that it is `false` , and MUST behave as described in Section 9: Continued Deposit
- The server MUST carry out authentication and mediated deposit as described in Section 8: Authentication and Mediated Deposit
- The server MUST create a new resource containing content equivalent to or derived from the deposited content, or return an error document.

Once the server has processed the request it SHOULD return a Deposit Receipt as described in Section 10: Deposit Receipt with the HTTP status code `201 (Created)` . The Deposit Receipt MUST be available under the **Edit-IRI** (Member Entry IRI), as supplied in the `Location` header.

For example:

```
201 Created
Location: [Edit-IRI]

[optional Deposit Receipt]
```

6.3.2. Creating a Resource with a Multipart Deposit

See [AtomMultipart] for details.

In order to ensure that all SWORD clients and servers can exchange a full range of file content and metadata, the use of Atom Multipart [AtomMultipart] is permitted to combine a package (possibly a simple ZIP) with a set of Dublin Core metadata terms [DublinCore] embedded in an Atom Entry.

The SWORD server is not required to support packaging formats, but this profile RECOMMENDS that the server be able to accept a ZIP file as the Media Part of an Atom Multipart request (See Section 5: IRIs and Section 7: Packaging for more details).

The client can create a resource by POSTing a multipart mime message to the **Col-IRI** with two parts: An Atom Entry containing the metadata for the deposit (known as the **Entry Part**), and the binary content as the second part (known as the **Media Part**), with the following requirements:

- The client MUST comply with the rules laid out in [AtomMultipart]
- The client MUST supply a `Content-Disposition` header of type `attachment` on the Entry Part with a `name` parameter set to `atom` [SWORD004]
- The client MUST supply a `Content-Disposition` header of type `attachment` on the Media Part with a `name` parameter set to `payload` and a `filename` parameter [SWORD004] (note that this requires the filename be expressed in ASCII).
- The client SHOULD supply a `Content-MD5` header with the MD5 checksum hex encoded for the binary content on the Media Part
- The client SHOULD supply a `Packaging` header [SWORD001] providing the IRI (or other allowed) of the packaging format used on the Media Part
- The client MAY provide an `In-Progress` header with a value of `true` or `false` [SWORD001] on the main HTTP header
- The client MAY provide an `On-Behalf-Of` header [SWORD001] on the main HTTP header
- The client MAY supply a `Slug` header providing a suggested identifier for the deposited content on the main HTTP header
- The client SHOULD add Dublin Core [DublinCore] terms to the Atom Entry as foreign markup (if appropriate); the terms MUST be embedded as direct children of the `atom:entry` element, if present.
- The client MAY add any other metadata formats or foreign markup to the `atom:entry` element

For example:

```
POST Col-IRI HTTP/1.1
Host: example.org
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
Content-Type: multipart/related;
            boundary="===============1605871705==";
            type="application/atom+xml"
In-Progress: true
On-Behalf-Of: jbloggs
Slug: [suggested identifier]
MIME-Version: 1.0

Media Post
--===============1605871705==
Content-Type: application/atom+xml; charset="utf-8"
Content-Disposition: attachment; name="atom"
MIME-Version: 1.0

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
        xmlns:dcterms="http://purl.org/dc/terms/">
    <title>Title</title>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2005-10-07T17:17:08Z</updated>
    <author><name>Contributor</name></author>
    <summary type="text">The abstract</summary>

    <!-- some embedded metadata -->
    <dcterms:abstract>The abstract</dcterms:abstract>
    <dcterms:accessRights>Access Rights</dcterms:accessRights>
    <dcterms:alternative>Alternative Title</dcterms:alternative>
    <dcterms:available>Date Available</dcterms:available>
    <dcterms:bibliographicCitation>Bibliographic Citation</dcterms:bibliographicCitation>
    <dcterms:contributor>Contributor</dcterms:contributor>
    <dcterms:description>Description</dcterms:description>
    <dcterms:hasPart>Has Part</dcterms:hasPart>
    <dcterms:hasVersion>Has Version</dcterms:hasVersion>
    <dcterms:identifier>Identifier</dcterms:identifier>
    <dcterms:isPartOf>Is Part Of</dcterms:isPartOf>
    <dcterms:publisher>Publisher</dcterms:publisher>
    <dcterms:references>References</dcterms:references>
```

```
    <dcterms:rightsHolder>Rights Holder</dcterms:rightsHolder>
    <dcterms:source>Source</dcterms:source>
    <dcterms:title>Title</dcterms:title>
    <dcterms:type>Type</dcterms:type>

</entry>
--===============1605871705==
Content-Type: application/zip
Content-Disposition: attachment; name=payload; filename=[filename]
Packaging: http://purl.org/net/sword/package/SimpleZip
Content-MD5: [md5-digest]
MIME-Version: 1.0

[...binary package data...]
--===============1605871705==--
```

The requirements placed on the server here are:

- The server SHOULD verify the `Content-MD5` header against the Media Part. If the check fails, the server MUST respond with 412 Precondition Failed. See Section 12: Error Documents.
- Server implementations MUST adopt the behaviour and requirements in [RFC2183] with respect to the `Content-Disposition` header for the Media Part
- If there is no `Packaging` header on the Media Part, the server SHOULD assume that it is the binary packaging format (See Section 5: IRIs)
- Server implementations MUST adopt the behaviour and requirements in Section 9.7 of [AtomPub] with respect to the `Slug` header
- If there is no `In-Progress` header, the server MUST assume that it is `false` , and MUST behave as described in Section 9: Continued Deposit
- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit
- The server MUST create a new resource containing content equivalent to or derived from the deposited content, or return an error document.
- The server MUST be able to record and later reflect the Dublin Core [DublinCore] foreign markup in the `atom:entry` element, but MAY NOT understand any other embedded formats.

Once the server has processed the request it SHOULD return a Deposit Receipt as described in Section 10: Deposit Receipt with the HTTP status code `201 (Created)` . The Deposit Receipt MUST be available under the **Edit-IRI** (Member Entry IRI), as supplied in the `Location` header.

For example:

```
201 Created
Location: [Edit-IRI]

[optional Deposit Receipt]
```

6.3.3. Creating a Resource with an Atom Entry

See Sections 5.3 and 9.2 of [AtomPub] for details.

Clients can create a container within a SWORD server and optionally provide it with metadata without adding any binary content to it. This is done by POSTing an Atom Entry to the **Col-IRI** with the following requirements:

- The client SHOULD supply a `Content-Type` header with the value `application/atom+xml;type=entry`
- The client MAY provide an `In-Progress` header with a value of `true` or `false` [SWORD001]. See Section 9: Continued Deposit for more details.
- The client MAY provide an `On-Behalf-Of` header [SWORD001]. See Section 8: Authentication and Mediated Deposit for more details.
- The client MAY supply a `Slug` header providing a suggested identifier for the deposited content
- The client SHOULD add Dublin Core [DublinCore] terms to the Atom Entry as foreign markup (if appropriate); the terms MUST be embedded as direct children of the `atom:entry` element, if present.
- The client MAY add any other metadata formats or foreign markup to the `atom:entry` element

For example:

```
POST Col-IRI HTTP/1.1
Host: example.org
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
Content-Type: application/atom+xml;type=entry
In-Progress: true
On-Behalf-Of: jbloggs
Slug: [suggested identifier]

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:dcterms="http://purl.org/dc/terms/">
    <title>Title</title>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
```

```
    <updated>2005-10-07T17:17:08Z</updated>
    <author><name>Contributor</name></author>
    <summary type="text">The abstract</summary>

    <!-- some embedded metadata -->
    <dcterms:abstract>The abstract</dcterms:abstract>
    <dcterms:accessRights>Access Rights</dcterms:accessRights>
    <dcterms:alternative>Alternative Title</dcterms:alternative>
    <dcterms:available>Date Available</dcterms:available>
    <dcterms:bibliographicCitation>Bibliographic Citation</dcterms:bibliographicCitation>
    <dcterms:contributor>Contributor</dcterms:contributor>
    <dcterms:description>Description</dcterms:description>
    <dcterms:hasPart>Has Part</dcterms:hasPart>
    <dcterms:hasVersion>Has Version</dcterms:hasVersion>
    <dcterms:identifier>Identifier</dcterms:identifier>
    <dcterms:isPartOf>Is Part Of</dcterms:isPartOf>
    <dcterms:publisher>Publisher</dcterms:publisher>
    <dcterms:references>References</dcterms:references>
    <dcterms:rightsHolder>Rights Holder</dcterms:rightsHolder>
    <dcterms:source>Source</dcterms:source>
    <dcterms:title>Title</dcterms:title>
    <dcterms:type>Type</dcterms:type>

</entry>
```

The requirements placed on the server here are:

- Server implementations MUST adopt the behaviour and requirements in Section 9.7 of [AtomPub] with respect to the `Slug` header
- If there is no `In-Progress` header, the server MUST assume that it is `false` , and MUST behave as described in Section 9: Continued Deposit
- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit
- The server MUST create a new resource, or return an error document.
- The server MUST be able to record and later reflect the Dublin Core [DublinCore] foreign markup in the `atom:entry` element, but MAY NOT understand any other embedded formats.
- The server MUST supply an **EM-IRI** for the client to use in future update operations, even though this EM-IRI may at this stage not return any content under HTTP GET.

Once the server has processed the request it SHOULD return a Deposit Receipt as described in Section 10: Deposit Receipt with the HTTP status code `201 (Created)` . The Deposit Receipt MUST be available under the **Edit-IRI** (Member Entry IRI), as supplied in the `Location` header.

For example:

```
201 Created
Location: [Edit-IRI]

[optional Deposit Receipt]
```

6.4. Retrieving the content

See section 5.4 of [AtomPub] for details.

The Deposit Receipt contains two IRIs which can be used to retrieve content from the server: **Cont-IRI** and **EM-IRI**. These are provided in the `atom:content@src` element and the `atom:link@rel="edit-media"` elements respectively. Their only functional difference is that the client MUST NOT carry out any HTTP operations other than GET on the **Cont-IRI**, while all operations are permitted on the **EM-IRI**. It is acceptable, but not required, that both IRIs to be the same, and in this section we refer only to the **EM-IRI** but in all cases it can be substituted for the **Cont-IRI**.

The Deposit Receipt contains zero or more `sword:packaging` elements [SWORD002], indicating to the client what package formats can be retrieved from the server. For example:

```
<entry xmlns="http://www.w3.org/2005/Atom"
        xmlns:sword="http://purl.org/net/sword/">

    [...]

    <content type="application/zip" src="http://swordapp.org/cont-IRI/43/my_deposit"/>
    <link rel="edit-media" href="http://swordapp.org/em-IRI/43/my_deposit"/>

    <sword:packaging>http://purl.org/net/sword/package/SimpleZip</sword:packaging>
    <sword:packaging>http://purl.org/net/sword/package/METSDSpaceSIP</sword:packaging>
    <sword:packaging>http://purl.org/net/sword/package/BagIt</sword:packaging>

</entry>
```

Requests to retrieve the Media Resource here refer to the Media Resource *as a whole*, as per [AtomPub], and not to any of the component parts of a complex Media Resource. That is, the response to a request for the Media Resource should

be considered a representation of the entire content of the object on the server, and it is not permissable to attempt to content negotiate for a particular part of the Media Resource. Implementers who wish to access the parts of a Media Resource should see Section 6.4.1. Atom Feed Representations of Media Resources below.

To retrieve the content in the desired packaging format, the client makes an HTTP GET request to the **EM-IRI** and MAY supply an `Accept-Packaging` header [SWORD001] with the IRI from one of the `sword:packaging` elements.

If the content is not in a publicly available space on the server, the user agent may be required to authenticate. In these cases the client MAY supply the `On-Behalf-Of` header [SWORD001] for informational purposes.

The server will respond in different ways depending on a number of conditions:

1. If the `Accept-Packaging` header is not supplied, the server SHOULD assume a simple ZIP packaging format (see Section 5: IRIs), but MAY elect another format if it wishes. The server MUST then supply the client with a package which meets these specifications, and SHOULD provide the SimpleZip IRI in a `Packaging` header on the response.
2. If the `Accept-Packaging` header is supplied, and contains the IRI (or other allowed value) of a format that the server supports, the server MUST then supply the client with a package which meets these specifications, and SHOULD provide an IRI (or other allowed value) of the format being returned in a `Packaging` header on the response.
3. If the `Accept-Packaging` header is supplied but contains a IRI (or other allowed value) for a format that the server does not support, the server MUST respond with 406 Not Acceptable and MAY include an error document as per Section 12: Error Documents.

Upon receiving a successful response, if no `Packaging` header is provided in the response the client SHOULD assume that it is a SimpleZip.

For more details about the packaging process, see Section 7: Packaging

6.4.1. Atom Feed Representations of Media Resources

See sections 11.2 and 12.1 of [AtomPub] for details.

It is RECOMMENDED that implementations provide an EM-IRI in the Deposit Receipt with a type of `application/atom+xml;type=feed` in addition to any other types available.

For example, the Depoisit Receipt would contain:

```
<entry xmlns="http://www.w3.org/2005/Atom"
        xmlns:sword="http://purl.org/net/sword/">

    [...]

    <link rel="edit-media" href="http://swordapp.org/em-IRI/43/my_deposit"/>
    <link rel="edit-media" type="application/atom+xml;type=feed" href="http://swordapp.org/em-IRI/43/my_deposit.feed"/>

    [...]

</entry>
```

The exact contents of the Feed representation are unspecified, but it is RECOMMENDED that this is a list of `atom:entry` elements which represent the individual media items which together form the Media Resource. Each `atom:entry` element MUST have an edit-media IRI which SHOULD be actionable as per Section 6.10.

NOTE that this is not the same as the Statement, which is a representation of the entire object on the server, not just the Media Resource. See Section 11: The SWORD Statement for details

6.5. Replacing the Content of a Resource

6.5.1. Replacing the File Content of a Resource

See section 5.4 and 9.3 of [AtomPub] for details.

The client can replace the existing content of a resource by performing an HTTP PUT of some new binary content to the **EM-IRI**, with the following requirements:

- The client SHOULD supply a `Content-Type` header
- The client MUST supply a `Content-Disposition` header with a `filename` parameter (note that this requires the filename be expressed in ASCII).
- The client SHOULD supply a `Content-MD5` header with the MD5 checksum hex encoded for the binary content
- The client SHOULD supply a `Packaging` header [SWORD001] providing the IRI (or other allowed) of the packaging format used
- The client MAY provide an `On-Behalf-Of` header [SWORD001]
- The client MAY provide a `Metadata Relevant` header [SWORD001] with the value `true` or `false`. This should be set to `true` if the server should consider the file a potential source of metadata extraction, or `false` if the server should not attempt to extract any metadata from the deposit.

For example:

```
PUT EM-IRI HTTP/1.1
Host: example.org
Content-Type: application/zip
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
Content-MD5: [md5-digest]
Content-Disposition: attachment; filename=[filename]
Packaging: http://purl.org/net/sword/package/METSDSpaceSIP
On-Behalf-Of: jbloggs

[request entity]
```

The requirements placed on the server here are:

- The server SHOULD verify the `Content-MD5` header against the content. If the check fails, the server MUST respond with 412 Precondition Failed. See Section 12: Error Documents.
- Server implementations MUST adopt the behaviour and requirements in [RFC2183] with respect to the `Content-Disposition` header
- If there is no `Packaging` header, the server SHOULD assume that it is the binary packaging format (See Section 5: IRIs)
- If the container to which the deposit is being made has been previously marked as "in progress", the server SHOULD continue to respect that assertion as per Section 9: Continued Deposit.
- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit.
- If no `Metadata-Relevant` header is provided the server MAY assume that it is `true` and MAY attempt to extract metadata from the deposit. If `Metadata-Relevant` is `false` the server SHOULD NOT attempt to extract metadata from the deposit.
- The server MUST effectively replace all the existing content in the item, although implementations may choose to provide versioning or some other mechanism for retaining the overwritten content.

Once the server has processed the request it MUST return an HTTP status code `204 (No Content)`, or an appropriate error code.

6.5.2. Replacing the Metadata of a Resource

See section 5.4 and 9.3 of [AtomPub] for details.

The client can replace the metadata of a resource by performing an HTTP PUT of a new Atom Entry on the **Edit-IRI**. The client can only be sure that the server will support this process when using the default format supported by SWORD: Qualified Dublin Core XML embedded directly in the `atom:entry`. Other metadata formats MAY be supported by a particular server, but this is not covered by the SWORD profile.

- The client SHOULD supply a `Content-Type` header with the value `application/atom+xml` or `application/atom+xml;type=entry`
- The client MAY provide an `On-Behalf-Of` header [SWORD001]
- The client MAY provide an `In-Progress` header with a value of true or false [SWORD001]. See Section 9: Continued Deposit for more details.

For example:

```
PUT Edit-IRI HTTP/1.1
Host: example.org
Content-Type: application/atom+xml
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
On-Behalf-Of: jbloggs

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:dcterms="http://purl.org/dc/terms/">
    <title>Title</title>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2005-10-07T17:17:08Z</updated>
    <author><name>Contributor</name></author>
    <summary type="text">The abstract</summary>

    <!-- some embedded metadata -->
    <dcterms:abstract>The abstract</dcterms:abstract>
    <dcterms:accessRights>Access Rights</dcterms:accessRights>
    <dcterms:alternative>Alternative Title</dcterms:alternative>
    <dcterms:available>Date Available</dcterms:available>
    <dcterms:bibliographicCitation>Bibliographic Citation</dcterms:bibliographicCitation>
    <dcterms:contributor>Contributor</dcterms:contributor>
    <dcterms:description>Description</dcterms:description>
    <dcterms:hasPart>Has Part</dcterms:hasPart>
    <dcterms:hasVersion>Has Version</dcterms:hasVersion>
    <dcterms:identifier>Identifier</dcterms:identifier>
    <dcterms:isPartOf>Is Part Of</dcterms:isPartOf>
    <dcterms:publisher>Publisher</dcterms:publisher>
    <dcterms:references>References</dcterms:references>
    <dcterms:rightsHolder>Rights Holder</dcterms:rightsHolder>
```

```
    <dcterms:source>Source</dcterms:source>
    <dcterms:title>Title</dcterms:title>
    <dcterms:type>Type</dcterms:type>

</entry>
```

The requirements placed on the server here are:

- If there is no `In-Progress` header, the server MUST assume that it is `false`, and MUST behave as described in Section 9: Continued Deposit
- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit.
- The server MUST effectively replace all the existing metadata in the item, although implementations may choose to provide versioning or some other mechanism for retaining the overwritten metadata.

Once the server has processed the request it MUST return an HTTP status code `200 (OK)` or `204 (No Content)`, or an appropriate error code.

6.5.3. Replacing the Metadata and File Content of a Resource

See [AtomMultipart] for details.

The client can replace both the metadata and content of a resource by performing an HTTP PUT on the **Edit-IRI** with a multipart mime message, as per Section 6.3.2. Creating a Resource with a Multipart Deposit

- The client MUST comply with the rules laid out in [AtomMultipart]
- The client MUST supply a `Content-Disposition` header of type `attachment` on the Entry Part with a `name` parameter set to `atom` [SWORD004]
- The client MUST supply a `Content-Disposition` header of type `attachment` on the Media Part with a `name` parameter set to `payload` and a `filename` parameter [SWORD004] (note that this requires the filename be expressed in ASCII).
- The client SHOULD supply a `Content-MD5` header with the MD5 checksum hex encoded for the binary content on the Media Part
- The client SHOULD supply a `Packaging` header [SWORD001] providing the IRI (or other allowed) of the packaging format used on the Media Part
- The client MAY provide an `In-Progress` header with a value of `true` or `false` [SWORD001] on the main HTTP header
- The client MAY provide an `On-Behalf-Of` header [SWORD001] on the main HTTP header
- The client SHOULD add Dublin Core [DublinCore] terms to the Atom Entry as foreign markup (if appropriate); the terms MUST be embedded as direct children of the `atom:entry` element, if present.
- The client MAY add any other metadata formats or foreign markup to the `atom:entry` element

For example:

```
PUT Edit-IRI HTTP/1.1
Host: example.org
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
Content-Type: multipart/related;
            boundary="===============1605871705==";
            type="application/atom+xml"
In-Progress: true
On-Behalf-Of: jbloggs
MIME-Version: 1.0

Media Post
--===============1605871705==
Content-Type: application/atom+xml; charset="utf-8"
Content-Disposition: attachment; name="atom"
MIME-Version: 1.0

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
        xmlns:dcterms="http://purl.org/dc/terms/">
    <title>Title</title>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2005-10-07T17:17:08Z</updated>
    <author><name>Contributor</name></author>
    <summary type="text">The abstract</summary>

    <!-- some embedded metadata -->
    <dcterms:abstract>The abstract</dcterms:abstract>
    <dcterms:accessRights>Access Rights</dcterms:accessRights>
    <dcterms:alternative>Alternative Title</dcterms:alternative>
    <dcterms:available>Date Available</dcterms:available>
    <dcterms:bibliographicCitation>Bibliographic Citation</dcterms:bibliographicCitation>
    <dcterms:contributor>Contributor</dcterms:contributor>
    <dcterms:description>Description</dcterms:description>
    <dcterms:hasPart>Has Part</dcterms:hasPart>
    <dcterms:hasVersion>Has Version</dcterms:hasVersion>
    <dcterms:identifier>Identifier</dcterms:identifier>
    <dcterms:isPartOf>Is Part Of</dcterms:isPartOf>
    <dcterms:publisher>Publisher</dcterms:publisher>
    <dcterms:references>References</dcterms:references>
    <dcterms:rightsHolder>Rights Holder</dcterms:rightsHolder>
```

```
    <dcterms:source>Source</dcterms:source>
    <dcterms:title>Title</dcterms:title>
    <dcterms:type>Type</dcterms:type>

</entry>
--===============1605871705==
Content-Type: application/zip
Content-Disposition: attachment; name=payload; filename=[filename]
Packaging: http://purl.org/net/sword/package/SimpleZip
Content-MD5: [md5-digest]
MIME-Version: 1.0

[...binary package data...]
--===============1605871705==--
```

The requirements placed on the server here are:

- The server SHOULD verify the `Content-MD5` header against the Media Part. If the check fails, the server MUST respond with 412 Precondition Failed. See Section 12: Error Documents.
- Server implementations MUST adopt the behaviour and requirements in [RFC2183] with respect to the `Content-Disposition` header for the Media Part
- If there is no `Packaging` header on the Media Part, the server SHOULD assume that it is the binary packaging format (See Section 5: IRIs)
- If there is no `In-Progress` header, the server MUST assume that it is `false`, and MUST behave as described in Section 9: Continued Deposit
- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit
- The server MUST overwrite the existing metadata and content with metadata and content equivalent to or derived from the deposited content, or return an error document. Servers may choose to implement versioning to retain previous copies of the content.
- The server MUST be able to record and later reflect the Dublin Core [DublinCore] foreign markup in the `atom:entry` element, but MAY NOT understand any other embedded formats.

Once the server has processed the request it MUST return an HTTP status code `200 (OK)` or `204 (No Content)`, or an appropriate error code.

6.6. Deleting the Content of a Resource

See Sections 5.4 and 9.4 of [AtomPub] for details.

The client can remove all the content of a resource without removing the resource itself by issuing an HTTP DELETE request on the **EM-IRI**.

After the delete operation, the container (represented by the **Edit-IRI**) must still have an **EM-IRI**, but it is an implementation decision as to whether to remove the old **EM-IRI** and provide a new one for subsequent addition of content, or whether to keep the original. This document RECOMMENDS keeping the original for simplicity.

The following rules apply to the client request:

- The client MAY provide an `On-Behalf-Of` header [SWORD001]

For example:

```
DELETE EM-IRI HTTP/1.1
Host: example.org
Authorization: Basic ZGFmZnk6c2VjZXJldA==
On-Behalf-Of: jbloggs
```

The requirements placed on the server here are:

- If the container to which the deposit is being made has been previously marked as "in progress", the server SHOULD continue to respect that assertion as per Section 9: Continued Deposit.
- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit.
- The server MUST effectively delete all the existing content in the item (subject to the rules associated with Metadata Handling), although implementations may choose to provide versioning or some other mechanism for retaining the deleted content, and may choose to re-use the **EM-IRI** (recommended).

Once the server has processed the request it MUST return an HTTP status code `204 (No Content)` upon success, or an appropriate error code.

6.7. Adding Content to a Resource

This section covers operations which allow you to add new files, packages and metadata to an existing resource on a SWORD server.

NOTE: [AtomPub] does not have a well-defined mechanism for adding new content to an Entry, and therefore does not clarify the difference between *new* content to be added to an existing resource and *alternative* content to be created as a

separate media resource (e.g. a translation of a document). This document explicitly does not attempt to deal with this issue.

6.7.1. Adding Content to the Media Resource

The client can add more content to the Media Resource itself by performing an HTTP POST request on the **EM-IRI**. This has the effect of placing new files into the Media Resource.

If the client wishes to add more content to an existing item on the server, it can send a package or an individual file to the **EM-IRI**. When the client sends an individual file (without a `Packaging` header), then the server will respond with an HTTP `Location` header which points to the location of the deposited resource. If a package is supplied, the `Location` header will refer to the Media Resource itself, and the client will need to refer to the Deposit Receipt for details as to the fate of the unpackaged content, or request an Atom Feed of the Media Resource (as per Section 6.4.1. Atom Feed Representations of Media Resources).

For files with extractable metadata, or packages which contain metadata relevant to the container, there is a `Metadata-Relevant` header which can be defined to indicate whether the deposit can be used to augment the metadata of the container.

Note that this section places tighter restrictions on the use of the `Packaging` header than other sections of this document, as there is increased scope for confusion here.

When carrying out this operation, the requirements for the client are:

- The client SHOULD supply a `Content-Type` header
- The client MUST supply a `Content-Disposition` header with a `filename` parameter (note that this requires the filename be expressed in ASCII).
- The client SHOULD supply a `Content-MD5` header with the MD5 checksum hex encoded for the binary content
- If sending a package, the client MUST supply a `Packaging` header [SWORD001] providing the IRI (or other allowed) of the packaging format used (absence of a `Packaging` header will be interpreted as an opaque file)
- The client MAY provide an `On-Behalf-Of` header [SWORD001]
- The client MAY provide a `Metadata-Relevant` header, with the value `true` or `false`. This should be set to `true` if the server should consider the file or package a potential source of metadata extraction, or `false` if the server should not attempt to extract any metadata from the deposit.

```
POST EM-IRI HTTP/1.1
Host: example.org
Content-Type: application/pdf
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
Content-MD5: [md5-digest]
Content-Disposition: attachment; filename=[filename]
On-Behalf-Of: jbloggs

[request entity]
```

The requirements placed on the server here are:

- The server SHOULD verify the `Content-MD5` header against the content. If the check fails, the server MUST respond with 412 Precondition Failed, and MAY return a SWORD Error document. See Section 12: Error Documents.
- Server implementations MUST adopt the behaviour and requirements in [RFC2183] with respect to the `Content-Disposition` header
- If there is no `Packaging` header, the server MUST assume that it is the binary packaging format (See Section 5: IRIs)
- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit.
- The server SHOULD NOT overwrite any existing content, and MUST add content equivalent to or derived from the deposited content to the existing container or respond with an error document.
- If there is no `Metadata-Relevant` header, the server MAY assume that it is `true` and MAY attempt to extract metadata from the deposit. If the `Metadata-Relevant` is `false` the server SHOULD NOT attempt to extract metadata from the deposit.

Once the server has processed the request it is RECOMMENDED that it returns a Deposit Receipt as described in Section 10: Deposit Receipt with the HTTP status code `201 (Created)`. The server MAY, though, return any response which it feels appropriate, and this profile does not limit the response types in any way. Nonetheless, the server MUST include an HTTP `Location` header with the IRI of the created resource (in the case of an individual file deposit) or the Media Resource itself (in the case of a package deposit). If an individual file has been deposited, the IRI of the created resource SHOULD be actionable as per Section 6.10: SWORD Actionable IRIs.

For example:

```
201 Created
Location: [File-IRI]

[optional Deposit Receipt]
```

6.7.2. Adding New Metadata to a Container

The client can update the metadata attached to a resource by issuing an HTTP POST of a new Atom Entry document containing metadata embedded as foreign markup to the **SE-IRI**.

Note: this section does not instruct the server on the best way to handle metadata, only that metadata SHOULD be added and not overwritten; in certain circumstances this may not produce the desired behaviour. Clients who wish to have full control over the metadata content should refer to Section 6.5.2. instead.

This operation is intended to allow the client to add metadata to an existing resource without overwriting existing metadata. This could be used, for example, by a self-claim system which allows authors to identify their items and to send to the server metadata pertaining to that particular user without affecting any of the other metadata already associated with the item. This operation is analagous to Section 6.3.3., although not all metadata sent in this case will necessarily be acted upon. Clients looking for precise control over the metadata record associated with an entry should refer to Section 6.5.2. instead.

The client must fulfill the following requirements:

- The client MUST supply a `Content-Type` header with the value `application/atom+xml` or `application/atom+xml;type=entry`
- The client MAY provide an `In-Progress` header with a value of `true` or `false` [SWORD001]
- The client MAY provide an `On-Behalf-Of` header [SWORD001].
- The client MAY add Dublin Core [DublinCore] terms to the Entry as foreign markup; the terms MUST be embedded as direct children of the `atom:entry` element.
- The client MAY add any other metadata format or foreign markup to the `atom:entry` element

```
POST SE-IRI HTTP/1.1
Host: example.org
Content-Type: application/atom+xml;type=entry
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
In-Progress: true
On-Behalf-Of: jbloggs

[entry document]
```

The requirements placed on the server here are:

- If there is no `In-Progress` header, the server MUST assume that it is `false` , and MUST behave as described in Section 9: Continued Deposit.
- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit.
- The server MUST be able to interpret and later reflect the Dublin Core [DublinCore] foreign markup in the `atom:entry` element, but MAY NOT understand any other embedded formats, but MUST NOT generate an error on the presence of not-understood markup.
- The server SHOULD only add metadata and not overwrite any existing metadata, but exact implementation will depend on the functions of the server. For example, if the server's metadata schema permits only a single instance a particular field, then the original value should be kept and the new value discarded; wheras if the schema permits multiple instances of a field the server may add the new values to the existing values without overwriting.

Once the server has processed the request it SHOULD return a Deposit Receipt as described in Section 10: Deposit Receipt with the HTTP status code `200 (OK)` . The Deposit Receipt MUST be available under the **Edit-IRI** (Member Entry IRI), which the server MAY supply in the `Location` header.

For example:

```
200 OK
Location: [Edit-IRI]

[optional Deposit Receipt]
```

6.7.3. Adding New Metadata and Packages or Files to a Resource with Multipart

The client can add new content and update the metadata attached to a resource by issuing an HTTP POST of an Atom Multipart [AtomMultipart] document to the **SE-IRI**.

Note: this section does not instruct the server on the best way to handle metadata or content: only that metadata SHOULD be added and not overwritten and that the content SHOULD be added to the existing Media Resource. If the client requires detailed control over metadata and content see Section 6.5.2 and Section 6.7.1 instead.

This operation is intended to allow the client to add metadata and content to an existing resource without overwriting existing metadata. The client would do this, perhaps, in a collaborative authoring environment when both files and metadata associated with them are coming from multiple sources. This operation is analagous to Section 6.3.2. except that the target IRI is the **EM-IRI** as the container already exists and may already contain content and metadata. Clients looking for detailed control over metadata and content should see Section 6.5.2 and Section 6.7.1 instead.

The client must fulfill the following requirements:

- The client MUST comply with the rules laid out in [AtomMultipart]
- The client MUST supply a `Content-Disposition` header of type `attachment` on the Entry Part with a `name` parameter set to `atom` [SWORD004]
- The client MUST supply a `Content-Disposition` header of type `attachment` on the Media Part with a `name` parameter set to `payload` and a `filename` parameter [SWORD004] (note that this requires the filename be expressed in ASCII).
- The client SHOULD supply a `Content-MD5` header with the MD5 checksum hex encoded for the binary content on the Media Part
- The client SHOULD supply a `Packaging` header [SWORD001] providing the IRI (or other allowed) of the packaging format used on the Media Part
- The client MAY provide an `In-Progress` header with a value of `true` or `false` [SWORD001] on the main HTTP header
- The client MAY provide an `On-Behalf-Of` header [SWORD001] on the main HTTP header
- The client MAY add Dublin Core [DublinCore] terms to the Entry as foreign markup; the terms MUST be embedded as direct children of the `atom:entry` element.
- The client MAY add any other metadata format to the `atom:entry` element
- The client MAY provide a `Metadata-Relevant` header, with the value `true`, although this is largely redundant (and is mentioned here only for a complete understanding of the behaviour the server will exhibit). A value of `false` is not permitted.

For example:

```
POST SE-IRI HTTP/1.1
Host: example.org
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
Content-Type: multipart/related;
      boundary="===============1605871705==";
      type="application/atom+xml"
In-Progress: true
On-Behalf-Of: jbloggs
Slug: [suggested identifier]
MIME-Version: 1.0

Media Post

--===============1605871705==
Content-Type: application/atom+xml; charset="utf-8"
Content-Disposition: attachment; name="atom"
MIME-Version: 1.0

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
        xmlns:dcterms="http://purl.org/dc/terms/">
    <title>Title</title>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2005-10-07T17:17:08Z</updated>
    <author><name>Contributor</name></author>
    <summary type="text">The abstract</summary>

    <!-- some embedded metadata -->
    <dcterms:abstract>The abstract</dcterms:abstract>
    <dcterms:accessRights>Access Rights</dcterms:accessRights>
    <dcterms:alternative>Alternative Title</dcterms:alternative>
    <dcterms:available>Date Available</dcterms:available>
    <dcterms:bibliographicCitation>Bibliographic Citation</dcterms:bibliographicCitation>
    <dcterms:contributor>Contributor</dcterms:contributor>
    <dcterms:description>Description</dcterms:description>
    <dcterms:hasPart>Has Part</dcterms:hasPart>
    <dcterms:hasVersion>Has Version</dcterms:hasVersion>
    <dcterms:identifier>Identifier</dcterms:identifier>
    <dcterms:isPartOf>Is Part Of</dcterms:isPartOf>
    <dcterms:publisher>Publisher</dcterms:publisher>
    <dcterms:references>References</dcterms:references>
    <dcterms:rightsHolder>Rights Holder</dcterms:rightsHolder>
    <dcterms:source>Source</dcterms:source>
    <dcterms:title>Title</dcterms:title>
    <dcterms:type>Type</dcterms:type>

</entry>
--===============1605871705==
Content-Type: application/zip
Content-Disposition: attachment; name=payload; filename=[filename]
Packaging: http://purl.org/net/sword/package/SimpleZip
Content-MD5: [md5-digest]
MIME-Version: 1.0

...binary package data...
--===============1605871705==--
```

The requirements placed on the server here are:

- The server SHOULD verify the `Content-MD5` header against the Media Part. If the check fails, the server MUST respond with 412 Precondition Failed. See Section 12: Error Documents.
- Server implementations MUST adopt the behaviour and requirements in [RFC2183] with respect to the `Content-Disposition` header for the Media Part.

- If there is no `Packaging` header on the Media Part, the server SHOULD assume that it is the binary packaging format (See Section 5: IRIs)
- Server implementations MUST adopt the behaviour and requirements in Section 9.7 of [AtomPub] with respect to the `Slug` header
- If there is no `In-Progress` header, the server MUST assume that it is `false` , and MUST behave as described in Section 9: Continued Deposit.
- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit.
- The server MUST be able to interpret the Dublin Core [DublinCore] foreign markup in the `atom:entry` element, but MAY NOT understand any other embedded formats, but MUST NOT generate an error on encountering not-understood foreign markup.
- The server SHOULD only add metadata and not overwrite any existing metadata, but exact implementation will depend on the functions of the server. For example, if the server's metadata schema permits only a single instance a particular field, then the original value should be kept and the new value discarded; wheras if the schema permits multiple instances of a field the server may add the new values to the existing values without overwriting.
- The server SHOULD add all of the content delivered in this operation to the pre-existing Media Resource, although it MAY take advantage of instructions in the packaging format as to how to structure the content appropriately; for example if the packaging format indicates multiple representations of the resource in different languages, these may be described as separate Media Resources with their `hreflang` attribute set appropriately in the Deposit Receipt
- If there is no `Metadata-Relevant` header, the server MUST assume that it is `true` and SHOULD use the Entry Part of the deposit as a source of metadata.

Once the server has processed the request it is RECOMMENDED that it returns a Deposit Receipt as described in Section 10: Deposit Receipt with the HTTP status code 201 (Created). The server MAY, though, return any response which it feels appropriate, and this profile does not limit the response types in any way. Nonetheless, the server MUST include an HTTP `Location` header with the IRI of the Media Resource itself (i.e. the **EM-IRI**).

For example:

```
201 Created
Location: [EM-IRI]

[optional Deposit Receipt]
```

6.8. Deleting the Container

The client can delete the entire object on the server by issuing an HTTP DELETE request on the **Edit-IRI**, with the following requirements:

- The client MAY provide an `On-Behalf-Of` header [SWORD001]

```
DELETE Edit-IRI HTTP/1.1
Host: example.org
Authorization: Basic ZGFmZnk6c2VjZXJldA==
On-Behalf-Of: jbloggs
```

The requirements placed on the server here are:

- If there is an `On-Behalf-Of` header, the server MUST behave as described in Section 8: Authentication and Mediated Deposit.
- The server MUST effectively remove the Container and all its content, although implementations may choose to provide versioning or some other mechanism for retaining the deleted content.
- The **Edit-IRI** MUST return a 404 Not Found on any future requests.

Once the server has processed the request is MUST respond with an HTTP status code `204 (No Content)` and leave the response body empty.

6.9. Retrieving the Statement

The Statement can be retrieved by requesting the IRI provided in the `atom:link@rel="http://purl.org/net/sword/terms/statement"` element. The format of the statement is identified by the `type` attribute of the `atom:link@rel="http://purl.org/net/sword/terms/statement"` element and MUST be `application/rdf+xml` for an OAI-ORE Resource Map, or `application/atom+xml;type=feed` for an Atom Feed.

It MAY also be possible to content negotiate for the statement on the **Edit-IRI**. All negotiable formats MUST also be available as `atom:link` elements with `@rel="http://purl.org/net/sword/terms/statement"` .

For details on these serialisations see Section 11: The SWORD Statement.

Servers MAY restrict access to the statement to authenticated users. In either case, the client MAY supply an `On-Behalf-Of` header for the purposes of indicating provenance to the server.

6.10. SWORD Actionable IRIs

In some cases the SWORD protocol will allow the server to provide IRIs to the client which refer to individual file resources (not the **EM-IRI**). This document RECOMMENDS that these IRIs are actionable to the extent that they support not only GET but also PUT and DELETE.

If these IRIs are not actionable as defined above then attempts to PUT or DELETE to them MUST result in a `403 (Forbidden)` or a `405 (Method Not Allowed)` HTTP response.

If these IRIs are actionable as defined above, they SHOULD support the headers as defined in Section 6.11. Responses to action requests MUST be the appropriate HTTP response codes.

6.11. Use of SWORD with arbitrary IRIs

During normal operation the SWORD server may respond to requests by the client with documents (such as the Statement) which contain IRIs for resources which are not formally covered by this profile. For example, the Statement may include IRIs to files unpacked from an original deposit package and made available independently. Or an implementation of a content management protocol such as CMIS [CMIS] or GData [GData] may provide IRIs for resources or collections of resources.

The following headers MAY be used by the client under the following conditions on any IRI outside of the scope of this profile, where the meanings of the headers are as defined in [SWORD001]:

Header	HTTP Method
Packaging	POST, PUT
Accept-Packaging	GET
On-Behalf-Of	GET, POST, PUT, DELETE
Metadata-Relevant	POST, PUT

The client MUST NOT assume that a SWORD server will support these headers for arbitrary IRIs. It is likely that the use of these headers by the client will be through explicit agreement between client and server implementation pairs that these headers will be supported alongside any content management API implemented in addition to SWORD.

7. Packaging

AtomPub uses the MIME mechanism to describe the data encoding for resources. This mechanism is inadequate where compound types are involved, such as tar archive files, METS packages, SCORM packages, MPEG21 DIDL packages etc. For example, a server might support certain METS profiles but not others. Other servers might be agnostic towards packaging, but support only certain contents within an archive.

In order to support package deposit and retrieval, SWORD uses the mechanisms described in [SWORD002] to extend AtomPub.

7.1. Package support in Service description

The SWORD Service Document uses the `sword:acceptPackaging` element from [SWORD002] to indicate that a SWORD collection will accept deposits of a particular packaging format. The client SHOULD assume that all packaging formats identified in this element apply to both binary and multipart deposits, and make reasonable requests based on this information. Content Negotiation is an inexact process where multiple "accept" fields are concerned, and it is RECOMMENDED that server implementations provide as coherent a set of rules as possible.

Clients should refer to the `sword:treatment` description in the Service Document to find out the treatment for a particular packaging type.

Packages formats SHOULD be identified by a IRI (as per [SWORD002]), but MAY be identified by an arbitrary string.

If no `sword:acceptPackaging` element is supplied the client MUST assume that the server does not formally support any package formats, and should expect everything to be treated as per the server's policies with regard to the mimetype as per the `app:accept` element.

```
<?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/net/sword/"
        xmlns:dcterms="http://purl.org/dc/terms/">

    <sword:version>2.0</sword:version>

    <workspace>

    <atom:title>Main Site</atom:title>
        <collection href="http://www.swordserver.org/atom/geography-collection" >
            <atom:title>My Repository : Geography Collection</atom:title>
            <accept>application/zip</accept>
            <sword:collectionPolicy>Collection Policy</sword:collectionPolicy>
            <dcterms:abstract>Collection description</dcterms:abstract>
            <sword:acceptPackaging>http://purl.org/net/sword/package/METSDSpaceSIP</sword:acceptPackaging>
            <sword:acceptPackaging>http://purl.org/net/sword/package/BagIt</sword:acceptPackaging>
            <sword:treatment>Treatment description</sword:treatment>
        </collection>
```

```
    </workspace>
</service>
```

7.2. Package support during resource creation

When creating Media Resources, the client SHOULD indicate the archive file MIME type using the HTTP `Content-Type` header, and SHOULD also give information about content packaging using the `Packaging` HTTP header [SWORD001].

The value of the `Packaging` header SHOULD match one of values the server has advertised as acceptable for the collection in the manner described in Section 7.1..

If a server receives a POST with an unacceptable `Packaging` header value, it MUST reject the POST by returning an HTTP response with a status code of `415 (Unsupported Media Type)` and a SWORD Error document with URI `http://purl.net/org/sword/terms/ErrorContent` (See Section 12: Error Documents), or store the content without further processing (as per [SWORD001]).

```
POST Col-IRI HTTP/1.1
Host: example.org
Content-Type: application/zip
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
Content-MD5: [md5-digest]
Content-Disposition: filename=myDSpaceMETSItem.zip
Packaging: http://purl.org/net/sword/package/METSDSpaceSIP
User-Agent: MyJavaClient/0.1 Restlet/2.0
```

7.3. Package description in Entry documents

When describing packaged resources in Media Entry documents, servers MUST use the `atom:content@type` attribute to describe the MIME type of the resource, and MAY add `sword:packaging` elements to the `atom:entry`.

If the server does not supply any `sword:packaging` elements, the client MUST assume that the server only supports a simple zip file packaging format (See Section 5: IRIs).

```
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:sword="http://purl.org/net/sword/">
    <title>My Deposit</title>
    <id>info:something:1</id>

    <updated>2008-08-18T14:27:08Z</updated>
    <author><name>jbloggs</name></author>
    <summary type="text">A summary</summary
    <generator uri="http://www.swordserver.org/engine" version="1.0"/>

    <content type="application/zip" src="http://swordapp.org/col-IRI/43/my_deposit/package.zip"/>
    <sword:packaging>http://purl.org/net/sword/package/METSDSpaceSIP</sword:packaging>
    <sword:packaging>http://purl.org/net/sword/package/BagIt</sword:packaging>
    <link rel="edit" href="http://swordapp.org/col-IRI/43/my_deposit.atom" />
</entry>
```

7.4. Negotiating for package formats

When retrieving a package identified in the `atom:content` element, the client MAY specify an `Accept-Packaging` HTTP header [SWORD001] indicating its desired package format. The value of this header SHOULD be one of the formats identified in the `sword:packaging` elements in the Deposit Receipt. If a packaging format is requested which is not available, the server MUST reject the GET by returning an HTTP response with a status code of `406 (Not Acceptable)` and a SWORD Error document with URI `http://purl.net/org/sword/terms/ErrorContent` (See Section 12: Error Documents).

For example:

```
GET /43/my_deposit/package.zip HTTP/1.1
Host: example.org
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: [content length]
Accept-Packaging: http://purl.org/net/sword/package/METSDSpaceSIP
User-Agent: MyJavaClient/0.1 Restlet/2.0
```

If there are multiple `atom:content` elements then it will not be clear which support which package formats. Servers are RECOMMENDED to provide as coherent a set of contents as possible.

8. Authentication and Mediated Deposit

In [AtomPub] Section 14, implementations MUST support HTTP Basic Authentication in conjunction with a TLS connection.

The SWORD Profile relaxes this requirement: SWORD client and server implementations SHOULD be capable of being configured to use HTTP Basic Authentication [RFC2617] in conjunction with a TLS connection as specified by [RFC2818].

In a number of situations the authenticated user using a SWORD client is not the owner of the deposited resource. SWORD provides a way of representing this usage by allowing clients to set a HTTP header field `On-Behalf-Of` which, if present SHOULD contain a user principle for the owner of the resource. It is also possible to use this SWORD mediation mechanism for situations such as non-interactive batch deposit in which the authenticated user is a software acting on behalf of a user.

The mediation technique described by SWORD is not intrinsically secure - it is assumed that trust between the owning user and the mediating user will be established by extending SWORD, or outside the scope of a resource creation, using a mechanism such as that described by [OAUTH].

8.1. Mediation In Service Description

SWORD servers SHOULD indicate whether they support mediated deposit by including a `sword:mediation` element containing a text element of either `true` or `false` in `app:collection` elements in Service Documents.

Clients SHOULD use the `On-Behalf-Of` header when retrieving Service Documents if they intend to use the feature for resource creation. Servers MAY use this header information along with authentication and any other information in constructing the Service Document representation returned. For example, the server might include only collections to which the `On-Behalf-Of` can deposit.

```
<?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/net/sword/"
        xmlns:dcterms="http://purl.org/dc/terms/">

    <sword:version>2.0</sword:version>

    <workspace>
        <atom:title>Main Site</atom:title>
        <collection href="http://swordapp.org/col-IRI/43" >
            <atom:title>Collection 43</atom:title>

            <accept>application/xml</accept>
            <accept>application/zip</accept>
            <accept>application/atom+xml</accept>

            <sword:collectionPolicy>Collection Policy</sword:collectionPolicy>
            <dcterms:abstract>Collection description</dcterms:abstract>

            <sword:mediation>true</sword:mediation>
            <sword:treatment>treatment description</sword:treatment>
            <sword:packaging>http://purl.org/net/sword/package/BagIt</sword:packaging>
        </collection>
    </workspace>
</service>
```

If the authentication credentials supplied fail, the server SHOULD respond with an HTTP status `401 (Unauthorized)`. If the authentication is successful but the user in the `On-Behalf-Of` header is not recognised the server SHOULD respond with an HTTP status `403 (Forbidden)` and SHOULD include a SWORD Error document with the URI `http://purl.org/net/sword/error/TargetOwnerUnknown` (see Section 12: Error Documents for details).

8.2. Recording Depositing Users

In all cases (mediated or not) where a user has authenticated to make a deposit, servers SHOULD preserve the user's identity in the `sword:depositedBy` property of the SWORD Statement. In mediated deposit, the value given in the `On-Behalf-Of` header SHOULD be used for the value of the `sword:depositedOnBehalfOf` property of the SWORD Statement. See Section 11: The SWORD Statement for details of the serialisation.

9. Continued Deposit

As scholarly systems may wish to give the client more control over the publishing process, SWORD uses the `In-Progress` HTTP header [SWORD001] to allow the client to indicate that a deposit should not yet be injected into any post-submission or pre-publication workflow. The `In-Progress` header MUST take the value `true` or `false`, and if it is not present the server MUST assume that it is `false` and behave as described below.

An example use case for this is that the client may be embedded into a system which uses the SWORD server as a storage layer, but which cannot acquire all of the content for a "finished" item in one deposit operation. Consider a user-facing system which encourages users to upload files one at a time through some web interface, which causes each file to be directly deposited onto the SWORD server. At the start of the deposit the client asserts that deposit is `In-Progress`, and then proceeds to upload files. If uploading them to the **SE-IRI** the client continues to assert `In-Progress: true` on each request (if depositing to **EM-IRI** this is not necessary). This goes on until the user confirms that they have uploaded all the relevant files, or navigates away from the page. At that stage, the client can issue a blank HTTP POST request to the SWORD server, with `In-Progress: false` to complete the deposit.

Note that the `In-Progress` header is intended to indicate to the server that further content will be coming in which is

associated with the existing content, before it can be considered "complete". It is not intended to provide workflow control, and clients MUST NOT assume that asserting `In-Progress: true` will have any specific effect on the state of the item.

9.1. Deposit Complete

If `In-Progress` is `false`, the server MUST assume that it can carry on processing the deposit or deletion as it sees fit.

9.2. Deposit Incomplete

If `In-Progress` is `true`, the server SHOULD expect the client to provide further updates to the item some undetermined time in the future. Details of how this is implemented is dependent on the server's purpose. For example, a repository system may hold items which are marked `In-Progress` in a workspace until such time as a client request indicates that the deposit is complete.

9.3. Completing a Previously Incomplete Deposit

The client can assert that a deposit process has completed by issuing an HTTP POST to the **SE-IRI** with a blank message body and with the `In-Progress` header set to false (it may simply omit the header altogether too, as this is treated as `In-Progress: false` by the server). The client SHOULD specify a `Content-Length: 0` HTTP header.

This operation is effectively equivalent to Section 6.7.2. Adding New Packages or Files to a Container, but with an empty POST body, and the client must fulfill the following requirements:

- The client MAY provide an `In-Progress` header with a value of `true` or `false` [SWORD001]
- The client MAY provide an `On-Behalf-Of` header [SWORD001]

```
POST SE-IRI HTTP/1.1
Host: example.org
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: 0
User-Agent: MyJavaClient/0.1 Restlet/2.0
In-Progress: false
```

If `In-Progress` is `false`, the server MUST assume that it can carry on processing the deposit or deletion as it sees fit. The server SHOULD NOT modify the existing content.

Once the server has processed the request it SHOULD return a Deposit Receipt as described in Section 10: Deposit Receipt with the HTTP status code `200 (OK)`. The Deposit Receipt MUST be available under the **Edit-IRI** (Member Entry IRI), which the server MAY supply in the `Location` header.

For example:

```
200 OK
Location: [Edit-IRI]

[optional Deposit Receipt]
```

10. Deposit Receipt

On successfully accepting a POST (deposit), implementers SHOULD return an Atom Entry Document with the HTTP response. The Atom Entry Document is known in SWORD as the Deposit Receipt must conform to the following conditions:

- It MAY contain information about the metadata associated with the deposit as it is stored on the server. If it does, it SHOULD provide this metadata in extended Dublin Core [DublinCore], although it MAY also provide it in other formats. Any Dublin Core markup should be in Dublin Core XML and embedded directly in the `atom:entry` element, not in a `dcterms:metadata` element.
- It MUST contain a Media Entry IRI (**Edit-IRI**), defined by `atom:link@rel="edit"`
- It MUST contain a Media Resource IRI (**EM-IRI**), defined by `atom:link@rel="edit-media"`
- It MUST contain a SWORD Edit IRI (**SE-IRI**), defined by `atom:link@rel="http://purl.org/net/sword/terms/add"` which MAY be the same as the **Edit-IRI**
- It MAY contain an arbitrary number of `sword:packaging` elements declaring the formats that the Media Resource can be retrieved in
- It MUST contain a single `sword:treatment` element [SWORD003] which contains either a human-readable statement describing treatment the deposited resource has received or a IRI that dereferences to such a description.
- It MAY contain a single `sword:verboseDescription` element [SWORD003] which should contain a verbose description of the deposit process. This MUST only be used for debugging and development purposes, and clients MUST NOT rely on this field to contain anything in particular.
- It MAY contain one or more `atom:link` element with the `rel` attribute of `http://purl.org/net/sword/terms/statement`, and the href to a IRI where the statement can be retrieved. The `atom:link` element MUST contain a type attribute which indicates the mime-type of the statement (for example `application/atom+xml;type=feed` or `application/rdf+xml`)
- It MAY contain an `atom:link@rel="alternate"` element with an `href` attribute which points to the splash page of the item on the server

- It SHOULD contain a single atom:link element with the `rel` attribute of `http://purl.org/net/sword/terms/originalDeposit` which provides the IRI identifying the package or file deposited in the current request. This information is transient and will only be present when responding to a deposit, it is not required that it be available in the Entry Document retreived from the EM-IRI.
- It SHOULD, if relevant, contain zero or more `atom:link` elements with the `rel` attribute of `http://purl.org.net/sword/terms/derivedResource` which provides the IRIs identifying any content resources which were derived or extracted from the originally deposited resource (e.g. the files unpacked from a zip file). These IRIs SHOULD be actionable as per Section 6.10: SWORD Actionable IRIs.

For example:

```
<entry xmlns="http://www.w3.org/2005/Atom"
        xmlns:sword="http://purl.org/net/sword/"
        xmlns:dcterms="http://purl.org/dc/terms/">

    <title>My Deposit</title>
    <id>info:something:1</id>
    <updated>2008-08-18T14:27:08Z</updated>
    <summary type="text">A summary</summary>
    <generator uri="http://www.myrepository.ac.uk/sword-plugin" version="1.0"/>

    <!-- the item's metadata -->
    <dcterms:abstract>The abstract</dcterms:abstract>
    <dcterms:accessRights>Access Rights</dcterms:accessRights>
    <dcterms:alternative>Alternative Title</dcterms:alternative>
    <dcterms:available>Date Available</dcterms:available>
    <dcterms:bibliographicCitation>Bibliographic Citation</dcterms:bibliographicCitation>
    <dcterms:contributor>Contributor</dcterms:contributor>
    <dcterms:description>Description</dcterms:description>
    <dcterms:hasPart>Has Part</dcterms:hasPart>
    <dcterms:hasVersion>Has Version</dcterms:hasVersion>
    <dcterms:identifier>Identifier</dcterms:identifier>
    <dcterms:isPartOf>Is Part Of</dcterms:isPartOf>
    <dcterms:publisher>Publisher</dcterms:publisher>
    <dcterms:references>References</dcterms:references>
    <dcterms:rightsHolder>Rights Holder</dcterms:rightsHolder>
    <dcterms:source>Source</dcterms:source>
    <dcterms:title>Title</dcterms:title>
    <dcterms:type>Type</dcterms:type>

    <sword:verboseDescription>Verbose description</sword:verboseDescription>
    <sword:treatment>Unpacked. JPEG contents converted to JPEG2000.</sword:treatment>

    <link rel="alternate" href="http://www.swordserver.ac.uk/col1/mydeposit.html"/>
    <content type="application/zip" src="http://www.swordserver.ac.uk/col1/mydeposit"/>
    <link rel="edit-media" href="http://www.swordserver.ac.uk/col1/mydeposit"/>
    <link rel="edit" href="http://www.swordserver.ac.uk/col1/mydeposit.atom" />
    <link rel="http://purl.org/net/sword/terms/add" href="http://www.swordserver.ac.uk/col1/mydeposit.atom" />
    <sword:packaging>http://purl.org/net/sword/package/BagIt</sword:packaging>

    <link rel="http://purl.org/net/sword/terms/originalDeposit"
            type="application/zip"
            href="http://www.swordserver.ac.uk/col1/mydeposit/package.zip"/>
    <link rel="http://purl.org/net/sword/terms/derivedResource"
            type="application/pdf"
            href="http://www.swordserver.ac.uk/col1/mydeposit/file1.pdf"/>
    <link rel="http://purl.org/net/sword/terms/derivedResource"
            type="application/pdf"
            href="http://www.swordserver.ac.uk/col1/mydeposit/file2.pdf"/>

    <link rel="http://purl.org/net/sword/terms/statement"
            type="application/atom+xml;type=feed"
            href="http://www.swordserver.ac.uk/col1/mydeposit.feed"/>
    <link rel="http://purl.org/net/sword/terms/statement"
            type="application/rdf+xml"
            href="http://www.swordserver.ac.uk/col1/mydeposit.rdf"/>


</entry>
```

11. The SWORD Statement

The Statement is a document which describes two features of the object as it appears on the server:

1. **Structure**. This may include originally uploaded content files, unpackaged content, derived content and any other features of the object
2. **State**. This allows the server to indicate to the client some information with regard to the state of the item on the server, including but not limited, to its ingest workflow position.

11.1. Predicates used by the Statement

All predicates used by the statement in any of its serialisations are defined here, and all exist under the sword namespace:

```
http://purl.org/net/sword/terms/
```

11.1.1. sword:originalDeposit

IRI: http://purl.org/net/sword/terms/originalDeposit

This is used to indicate the IRI of a resource on the server which was an original deposit package. This allows the client to easily identify any packages which form the basis of the original deposit among other parts of the object which may be derived from the original.

In XML this may be serialised as:

```
<sword:originalDeposit href="http://location/of/deposit"/>
```

In RDF/XML this can be serialised as as:

```
<sword:originalDepsit rdf:resource="http://location/of/deposit"/>
```

11.1.2. sword:state

IRI: http://purl.org/net/sword/terms/state

This is used to supply the IRI representing the state of an item on the server. This can be any IRI, and is not constrained to any ontology. This can be used in combination with `sword:stateDescription` to provide more details to the client.

In XML this may be serialised as:

```
<sword:state href="http://sword/state"/>
```

In RDF/XML this can be serialised as:

```
<sword:state rdf:resource="http://sword/state" />
```

11.1.3. sword:packaging

IRI: http://purl.org/net/sword/terms/packaging

This is used to identify the packaging format of a resource. Particularly, if used in conjunction with `sword:originalDeposit`, this can be used to indicate the package format that an original deposit package used.

In XML this may be serialised as:

```
<sword:originalDeposit href="http://location/of/deposit">
    <sword:packaging>http://sword/packaging/</sword:packaging>
</sword:originalDeposit>
```

In RDF/XML this can be serialised as:

```
<sword:originalDepsit rdf:resource="http://location/of/deposit"/>
<rdf:Description rdf:about="http://location/of/deposit">
    <sword:packaging rdf:resource="http://sword/packaging/"/>
</rdf:Description>
```

11.1.4. sword:depositedOn

IRI: http://purl.org/net/sword/terms/depositedOn

This is used to provide a user who was responsible for depositing an original sword package. The use credentials may just be the username of the depositor, but the server is free to add further details where necessary.

In XML this may be serialised as:

```
<sword:originalDeposit href="http://location/of/deposit">
    <sword:depositedOn>2011-01-01T00:00:00Z</sword:depositedOn>
</sword:originalDeposit>
```

In RDF/XML this may be serialised as

```
<sword:originalDepsit rdf:resource="http://location/of/deposit"/>
<rdf:Description rdf:about="http://location/of/deposit">
    <sword:depositedOn rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2011-01-01T00:00:00Z</sword:depositedOn>
</rdf:Description>
```

11.1.5. sword:depositedBy

IRI: http://purl.org/net/sword/terms/depositedBy

This is used to provide a user who was responsible for depositing an original sword package. The use credentials may just be the username of the depositor, but the server is free to add further details where necessary.

In XML this may be serialised as:

```
<sword:originalDeposit href="http://location/of/deposit">
    <sword:depositedBy>jbloggs</sword:depositedBy>
</sword:originalDeposit>
```

In RDF/XML this may be serialised as

```
<sword:originalDepsit rdf:resource="http://location/of/deposit"/>
<rdf:Description rdf:about="http://location/of/deposit">
        <sword:depositedBy>jbloggs</sword:depositedBy>
</rdf:Description>
```

11.1.6. sword:depositedOnBehalfOf

IRI: http://purl.org/net/sword/terms/depositedOnBehalfOf

This is used to provide the user for whom the original sword package was deposited on behalf of. The use credentials may just be the username of the depositor, but the server is free to add further details where necessary.

In XML this may be serialised as:

```
<sword:originalDeposit href="http://location/of/deposit">
    <sword:depositedOnBehalfOf>jbloggs</sword:depositedOnBehalfOf>
</sword:originalDeposit>
```

In RDF/XML this may be serialised as

```
<sword:originalDepsit rdf:resource="http://location/of/deposit"/>
<rdf:Description rdf:about="http://location/of/deposit">
    <sword:depositedOnBehalfOf>jbloggs</sword:depositedOnBehalfOf>
</rdf:Description>
```

11.1.7. sword:stateDescription

IRI: http://purl.org/net/sword/terms/stateDescription

This is used to provide a human readable description of the state that the item is in on the server. This is used in conjunction with the `sword:state` to provide additional details for the client to use in its interface with the end user.

In XML this may be serialised as:

```
<sword:state href="http://sword/state">
        <sword:stateDescription>The item has passed through the workflow and is now archived</sword:stateDescription>
</sword:state>
```

In RDF/XML this may be serialised as

```
<sword:state rdf:resource="http://sword/state"/>
<rdf:Description rdf:about="http://sword/state">
        <sword:stateDescription>The item has passed through the workflow and is now archived</sword:stateDescription>
</rdf:Description>
```

11.2. Statement Requirements

This section details the requirements of a Statement document. For specific details as to how to implement these in the

various serialisations, see later sections.

- The Statement SHOULD identify any original deposit packages with the `sword:originalDeposit` term
- The Statement MAY specify one or more states for the item on the server with the `sword:state` term
- If present, the `sword:state` term SHOULD include a human readable description in the `sword:stateDescription` term
- If present, the `sword:originalDeposit` term MAY include one or more `sword:packaging` terms
- If present, the `sword:originalDeposit` term MAY include the date the package was originally deposited in the `sword:depositedOn` term
- If present, the `sword:originalDeposit` term MAY include the depositor of the package in the `sword:depositedBy` term
- If present, the `sword:originalDeposit` term MAY include the user on whose behalf the deposit was made in the `sword:depositedOnBehalfOf` term

The client's responsibilities are:

- If no `sword:packaging` term is provided on `sword:originalDeposit` , the client MUST assume only a simple zip packaging format (See Section 5: IRIs).

11.3. OAI-ORE Serialisation

The SWORD statement can be provided as an OAI-ORE Resource [OAIORE] Map with SWORD extensions. For an RDF document to comply with the requirements of the SWORD statement it needs to meet the following requirements but is not constrained in any way as to what additional data is provided alongside. As such, it is expected that the SWORD statement may be included with existing RDF serialisations of objects provided by the server.

The responsibilities of the server are (as serialisation-specific versions of the requirements listed in 11.2. Statement Requirements):

- The aggregation SHOULD identify any original deposit packages with the `sword:originalDeposit` term
- The aggregation MAY specify one or more states for the item on the server with the `sword:state` term
- If present, the `sword:state` term SHOULD include a human readable description in the `sword:stateDescription` term
- If present, the `sword:originalDeposit` term MAY include one or more `sword:packaging` terms
- If present, the `sword:originalDeposit` term MAY include the date the package was originally deposited in the `sword:depositedOn` term
- If present, the `sword:originalDeposit` term MAY include the depositor of the package in the `sword:depositedBy` term
- If present, the `sword:originalDeposit` term MAY include the user on whose behalf the deposit was made in the `sword:depositedOnBehalfOf` term
- The statement MAY include any other ORE terms, and any other terms from other ontologies
- All individual files which are listed as `ore:aggregation` elements SHOULD be actionable as per Section 6.10: SWORD Actionable IRIs.

For example:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ore="http://www.openarchives.org/ore/terms/">

    <rdf:Description rdf:about="http://localhost:8080/edit-IRI/43/my_deposit">
        <ore:describes rdf:resource="http://localhost:8080/agg-IRI/43/my_deposit"/>
    </rdf:Description>

    <rdf:Description rdf:about="http://localhost:8080/agg-IRI/43/my_deposit">
        <ore:isDescribedBy rdf:resource="http://localhost:8080/edit-IRI/43/my_deposit"/>
        <ore:aggregates rdf:resource="http://localhost:8080/part-IRI/43/my_deposit/example.zip"/>
        <sword:originalDeposit rdf:resource="http://localhost:8080/part-IRI/43/my_deposit/example.zip"/>
        <sword:state rdf:resource="http://purl.org/net/sword/state/archived"/>
    </rdf:Description>

    <rdf:Description rdf:about="http://localhost:8080/part-IRI/43/my_deposit/example.zip">
        <sword:packaging rdf:resource="http://purl.org/net/sword/package/SimpleZip"/>
        <sword:depositedOn rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
            2011-02-25T14:40:09Z
        </sword:depositedOn>
        <sword:depositedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
            sword
        </sword:depositedBy>
    </rdf:Description>

    <rdf:Description rdf:about="http://purl.org/net/sword/state/archived">
        <sword:stateDescription>
            The work has passed through review and is now in the archive
        </sword:stateDescription>
    </rdf:Description>

</rdf:RDF>
```

The client's responsibilities are (in addition to those specified in 11.2. Statement Requirements):

- The client MUST ignore without error any terms in the document that it does not understand
- The client SHOULD recognise parts of the resource map which are not identified with `sword:originalDeposit` terms as content of the item, and represent them appropriately

11.4. Atom Serialisation

The SWORD statement can be provided as an Atom Feed [Atom] with sword extensions. For an Atom Feed document to comply with the requirements of the SWORD statement it needs to meet the following requirements but is not constrained in any way as to what additional data is provided alongside. As such, it is expected that the SWORD statement may be included with existing serialisations provided by the server, such as with any GData [GData] extensions.

The responsibilities of the server are (as serialisation-specific versions of the requirements listed in 11.2. Statement Requirements):

- The Feed MUST represent files contained in the item as an `atom:entry` element (this does not mandate that all files in the item are listed, though)
- Each `atom:entry` which is an original deposit file MUST have an `atom:category` element with the term `sword:originalDeposit` (this does not mandate that all original deposits are listed as entries)
- The Feed MAY specify one more more states for the item on the server with an `atom:category` element with the term `http://purl.org/net/sword/terms/state` root `atom:feed` element
- If present, the `sword:state` category term SHOULD include a human readable description in the body of the `atom:category` element.
- Each `atom:entry` which is an original deposit MAY include one or more `sword:packaging` terms as foreign markup in the `atom:entry`
- Each `atom:entry` which is an original deposit MAY include the date the package was originally deposited in the `sword:depositedOn` term as foreign markup in the `atom:entry`
- Each `atom:entry` which is an original deposit MAY include the depositor of the package in the `sword:depositedBy` term as foreign markup in the `atom:entry`
- Each `atom:entry` which is an original deposit MAY include the user on whose behalf the deposit was made in the `sword:depositedOnBehalfOf` term as foreign markup in the `atom:entry`
- The statement MAY include any other foreign markup and legitimate atom extensions

For example:

```
<atom:feed xmlns:sword="http://purl.org/net/sword/terms/"
           xmlns:atom="http://www.w3.org/2005/Atom">

    <atom:category scheme="http://purl.org/net/sword/terms/state"
        term="[state identifier]"
        label="State">
            The work has passed through review and is now in the archive
    </atom:category>

    <atom:entry>
        <atom:category scheme="http://purl.org/net/sword/terms/"
                term="http://purl.org/net/sword/terms/originalDeposit"
                label="Orignal Deposit"/>
        <atom:content type="application/zip"
                    src="http://localhost:8080/part-IRI/43/my_deposit/example.zip"/>
        <sword:packaging>http://purl.org/net/sword/package/SimpleZip</sword:packaging>
        <sword:depositedOn>2011-03-02T20:50:06Z</sword:depositedOn>
        <sword:depositedBy>sword</sword:depositedBy>
        <sword:depositedOnBehalfOf>jbloggs</sword:depositedBy>
    </atom:entry>

</atom:feed>
```

The client's responsibilities are (in addition to those specified in 11.2. Statement Requirements):

- The client MUST ignore without error any terms in the document that it does not understand

12. Error Documents

SWORD adds a new class of document to [AtomPub] that gives server implementations the ability to describe error conditions more fully than HTTP response codes allow, as per [SWORD003]. If a server is reporting an error condition in response to a resource POST, it SHOULD also return a document with a root element of sword:error. The sword:error element SHOULD have an href attribute containing a IRI that identifies the error. Errors in the SWORD namespace are reserved and legal values are enumerated below. Implementations MAY define their own errors, but MUST use a different namespace to do so.

The `sword:error` element MAY contain any of the elements normally used in the Deposit Receipt, but all fields are OPTIONAL.

The error document SHOULD contain an `atom:summary` element with a short description of the error.

The error document MAY contain a `sword:verboseDescription` element with a long description of the problem or any other appropriate software-level debugging output (e.g. a stack trace). Server implementations may wish to provide this for client developers' convenience, but may wish to disable such output in any production systems.

The server SHOULD specify that the `Content-Type` of the is `text/xml` or `application/xml` .

12.1. Error URIs

12.1.1. sword:ErrorContent

IRI: http://purl.org/net/sword/error/ErrorContent

The supplied format is not the same as that identified in the `Packaging` header and/or that supported by the server

Associated HTTP Status: 415 (Unsupported Media Type) or 406 (Not Acceptable)

12.1.2. sword:ErrorChecksumMismatch

IRI: http://purl.org/net/sword/error/ErrorChecksumMismatch

Checksum sent does not match the calculated checksum. The server MUST also return a status code of `412 Precondition Failed`

12.1.3. sword:ErrorBadRequest

IRI: http://purl.org/net/sword/error/ErrorBadRequest

Some parameters sent with the POST were not understood. The server MUST also return a status code of `400 Bad Request` .

12.1.4. sword:TargetOwnerUnknown

IRI: http://purl.org/net/sword/error/TargetOwnerUnknown

Used in mediated deposit when the server does not know the identity of the `On-Behalf-Of` user.

Associated HTTP Status: 403 (Forbidden)

12.1.5. sword:MediationNotAllowed

IRI: http://purl.org/net/sword/error/MediationNotAllowed

Used where a client has attempted a mediated deposit, but this is not supported by the server. The server MUST also return a status code of `412 Precondition Failed` .

12.1.6. sword:MethodNotAllowed

IRI: http://purl.org/net/sword/error/MethodNotAllowed

Used when the client has attempted one of the HTTP update verbs (POST, PUT, DELETE) but the server has decided not to respond to such requests on the specified resource at that time. The server MUST also return a status code of `405 Method Not Allowed`

12.1.7. sword:MaxUploadSizeExceeded

IRI: http://purl.org/net/sword/error/MaxUploadSizeExceeded

Used when the client has attempted to supply to the server a file which exceeds the server's maximum upload size limit

Associated HTTP Status: 413 (Request Entity Too Large)

12.2. Error Document Example

```
HTTP 1.1  400 Bad Request

<?xml version="1.0" encoding="utf-8"?>
<sword:error xmlns="http://www.w3.org/2005/Atom"
       xmlns:sword="http://purl.org/net/sword/"
       xmlns:arxiv="http://arxiv.org/schemas/atom"
       href="http://example.org/errors/BadManifest">
    <author>
        <name>Example repository</name>
    </author>
    <title>ERROR</title>
    <updated>2008-02-19T09:34:27Z</updated>

    <generator uri="https://example.org/sword-app/"
               version="0.9">sword@example.org</generator>

    <summary>The manifest could be parsed, but was not valid -
    no technical metadata was provided.</summary>
    <sword:treatment>processing failed</sword:treatment>
    <sword:verboseDescription>
        Exception at [ ... ]
    </sword:verboseDescription>
    <link rel="alternate" href="https://arxiv.org/help" type="text/html"/>

</sword:error>
```

13. Auto-Discovery

AtomPub makes no recommendations on the discovery of Service Documents or other resources. SWORD RECOMMENDS that server implementations use the following conventions

13.1. For Service Documents

Embed an html link with a rel value of `sword` (for backwards compatibility with SWORD 1.3 recommendations), or a rel value of `http://purl.org/net/sword/terms/service-document`

```
<html:link rel="sword" href="[Service Document URL]"/>
```

or

```
<html:link rel="http://purl.org/net/sword/discovery/service-document" href="[Service Document URL]"/>
```

13.2. For Deposit Endpoints

Embed an html link with a rel value of `http://purl.org/net/sword/terms/deposit` in any page which represents a deposit endpoint (for example, the splash page for a Collection)

```
<html:link rel="http://purl.org/net/sword/terms/deposit" href="[Deposit URL]"/>
```

13.3. For Resources

Embed an html link with a rel value of `http://purl.org/net/sword/terms/edit` in any page which represents a deposited resource, and optionally include a `http://purl.org/net/sword/terms/statement` link as well.

```
<html:link rel="http://purl.org/net/sword/terms/edit" href="[Edit-IRI]"/>
<html:link rel="http://purl.org/net/sword/terms/statement" href="[State-IRI]"/>
```

14. References

[Atom] Nottingham, M. and R. Sayre, "The Atom Syndication Format", RFC 4287, December 2005. http://www.ietf.org/rfc/rfc4287.txt

[AtomMultipart] Gregario, J, et al "AtomPub Multipart Media Resource Creation", September 2008. http://tools.ietf.org/html/draft-gregorio-atompub-multipart-04

[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)

[CMIS] Content Management Interoperability Services (CMIS) Version 1.0. http://docs.oasis-open.org/cmis/CMIS/v1.0/cs01/cmis-spec-v1.0.html

[DublinCore] DCMI Metadata Terms. http://www.dublincore.org/documents/dcmi-terms/

[GData] Google Documents List API. http://code.google.com/apis/documents/docs/developers_guide.html

[OAUTH] Atwood, A., Conlan, R. et al OAuth Core 1.0 http://oauth.net/core/1.0/, December 2007

[OAIORE] Lagoze, C., Van de Sompel, H et al, Open Archives Initiative Object Reuse and Exchange, http://www.openarchives.org/ore/, June 2008

[RDF] Resource Description Framework. http://www.w3.org/RDF/

[RFC2119] Bradner, S. "Key words for use in RFCs to Indicate Requirement Levels". http://www.ietf.org/rfc/rfc2119.txt, March 1997.

[RFC2183] Troost, R., Dorner, S. and Moore, K., "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, August 1997 http://www.ietf.org/rfc/rfc2183.txt

[RFC2616] Fielding et al, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999 http://www.w3.org/Protocols/rfc2616/rfc2616.html

[RFC2617] Franks, J. et al, "HTTP Authentication: Basic and Digest Access Authentication". http://www.ietf.org/rfc/rfc2617.txt, June 1999

[RFC2818] Rescorla, E. "HTTP over TLS". http://www.ietf.org/rfc/rfc2818.txt, May 2000.

[SWORD 1.3] Downing, J. "SWORD AtomPub Profile version 1.3", 2008. http://www.swordapp.org/docs/sword-profile-

1.3.html

[SWORD001] Jones, R. "Packaged Content Delivery over HTTP", February 2011

[SWORD002] Jones, R. "AtomPub Extensions for Packaged Content", February 2011

[SWORD003] Jones, R. "AtomPub Extensions for Scholarly Systems", February 2011

[SWORD004] Jones, R. "Atom Multipart Extensions for Packaged Content", February 2011