An overview of the DSpace import and export functionality

1. Using DSpace Import Tool

Command line tools

DSpace has a set of command line tools for importing and exporting items in batches, using the DSpace simple archive format. The procedure for importing the items consists of a few simple steps:

- Create the records using the DSpace Archive Directory format
- Import the Archive Directory using the command line tool

DSpace Archive Directory

The basic concept behind the DSpace's simple archive format is to create an archive, which is a directory full of items, with a subdirectory per item. Each item directory contains a file for the item's descriptive metadata, a contents files that lists the bitstreams for the item, and of course the files that make up the bitstreams.

```
archive_directory/
  item_000/
    dublin_core.xml          - qualified Dublin Core metadata for metadata fields
    metadata_[prefix].xml    - metadata in another schema, the prefix is the name
                               of the schema as registered with the metadata registry
    contents                 - text file containing one line per filename
    file_1.doc               - files to be added as bitstreams to the item
    file_2.pdf
  item_001/
    dublin_core.xml
    contents
    file_1.png
    ...
```

Metadata

The `dublin_core.xml` or `metadata_[prefix].xml` file has the following format, where each metadata element has it's own entry within a `<dcvalue>` tagset. There are currently three tag attributes available in the `<dcvalue>` tagset:

- `element`: the Dublin Core element
- `qualifier`: the element's qualifier
- `language`: (optional) ISO language code for element

An example dublin_core.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<dublin_core>
  <dcvalue element="title" qualifier="none">A Tale of Two Cities</dcvalue>
  <dcvalue element="date" qualifier="issued">1990</dcvalue>
  <dcvalue element="title" qualifier="alternate" language="fr">J'aime les Printemps</
dcvalue>
</dublin_core>
```

Every metadata field used, must be registered via the metadata registry of the DSpace instance first.

Files

The `contents` file simply enumerates, one file per line, the bitstream file names. See the following example:

```
        file_1.doc
        file_2.pdf
        license.txt
```


Notice that the license is optional, and that items can refer to a single license in one of the items. To do so the license file can be placed in item_000 and other content files can reference it using relative paths, for example `.../item_000/`.

The bitstream name may optionally be followed by the sequence:

```
\tbundle:bundlename
```

where '`\t`' is the tab character and `bundlename` is replaced by the name of the bundle to which the bitstream should be added. If no bundle is specified, the bitstream will be added to the 'ORIGINAL' bundle.

The bitstream name may optionally be followed by the sequence:

```
\tpermissions:r-'groupname'
```

where '`\t`' is the tab character, either the character '`r`' or the character '`w`' is used to determine whether read or write permissions are being configured and '`groupname`' is replaced by the name of the group for which the permissions should be configured. If no permissions are specified, the default collection permissions will be applied to the uploaded file. Multiple permissions can be added by specifying multiple permissions blocks.

The bitstream name may optionally be followed by the sequence:

```
\tpermissions:-r 'groupname'
```

where '`\t`' is the tab character and '`descriptive text`' is replaced by the text which will be assigned to the bitstream.

An example `contents` file including optional parameters is:

```
img1.png  permissions:-r 'Anonymous'  description:descriptivetext
img2.png  bundle:thumbnail
```

Import Command

Basic command

The import command will always start with `[dspace]/bin/import` and followed by a number of the parameters specified below.

Example: To add items to a collection with an EPerson as the submitter:

```
[dspace]/bin/import --add --eperson=joe@user.com --collection=collectionID
                    --source=items_dir --mapfile=mapfile
```

which would cycle through the archive directory's items, import them, and then generate a **mapfile** which stores the mapping of item directories to item handles. Save this map file! Using the map file you can then 'unimport' with the command:

```
[dspace]/bin/import --delete --mapfile=mapfile
```

Parameters

Action

Exactly one action must be specified:

```
--add
```

Implies all items in the archive directory are added as new items in the DSpace repository

`--replace`

Implies all items in the archive directory having a corresponding line in the map file will be updated, while the items in the archive directory without a corresponding line in the mapping file are added as new items in the DSpace repository

`--delete`

Implies all items the DSpace repository corresponding to a line in the mapping file will be withdrawn from the repository

Input specification

`--source=items_dir`

Specifies the path to the archive directory which has been created before. The source is required when either the add or the replace action is used.

`--collection=collectionID`

Specifies the collection(s) to which the items should all be added. The value can either be the internal collection id, or the collection handle (e.g. 123456789/35). The collection handle is a more useful format as this can be determined easily for each collection based on the URL in the browser. The collection is required when either the add or the replace action is used.

`--mapfile=mapfile`

Specifies the file containing the mapping between records in the archive directory and DSpace items. The mapfile is always required.

`--eperson=joe@user.com`

Specifies the DSpace Eperson to be attached as the submitter for the imported items. DSpace requires a submitter for each item. The submitter must have the required permissions for executing the action. Usually a general administrator account will be used for this task. The eperson is always required.

`--workflow`

If this parameter is added, the standard collection workflow will be used for all imported items. This implies that the metadata editors configured for the specified collection will receive a task for each of the imported items. If this parameter is omitted, the imported will be archived immediately.

`--test`

If this parameter is added, a test import will be executed. This implies that the metadata will be read, but not actually imported in the repository. A test import can be executed to find errors in the archive directory or missing parameters before executing an actual import. If this parameter is omitted, the actual import will be executed.

`--template`

If this parameter is added, the collection item template will be applied. The items will be imported starting with the metadata present in the item template, and the metadata present in the metadata xml files will be added. If this parameter is omitted, the item template will be ignored and the imported items will only contain the metadata present in the metadata xml files.

`--resume`

If this parameter is used, the mapfile will be used to determine which items were previously added to the repository. The remaining items will be imported into the repository. This is especially useful for failed imports.

Exercises

1.1. An example archive directory can be downloaded from http://atmire.com/courses/archive_directory.zip. Download and unzip this directory and place it in your dspace directory. Review the contents of this archive directory to understand what will happen when importing this archive directory. Import this directory into a collection of your choosing.

1.2. Add one additional item, having arbitrary metadata and files connected to them, and perform a resumed import in the same collection. After importing, check whether this did not import the previously items imported again, but has only imported the new items.

2. DSpace package ingestion / dissemination

Command line tools

This command-line tool gives you access to the Packager plugins. It can ingest a package to create a new DSpace Item, or disseminate an Item as a package. Contrary to the Import Tool, the packager is used for importing or exporting a single item.

Ingesting

To ingest a package from a file, give the command:

```
[dspace]/bin/packager -e user -c handle -t packager path
```

Where user is the e-mail address of the E-Person under whose authority this runs; handle is the Handle of the collection into which the Item is added, packager is the plugin name of the package ingester to use, and path is the path to the file to ingest (or "-" to read from the standard input).

Here is an example that loads a PDF file with *internal metadata* as a package:

```
/dspace/bin/packager -e florey@mit.edu -c 1721.2/13 -t pdf thesis.pdf
```

This example takes the result of retrieving a URL and ingests it:

```
wget -O - http://alum.mit.edu/jarandom/my-thesis.pdf | \

/dspace/bin/packager -e florey@mit.edu -c 1721.2/13  -t pdf -
```

Disseminating

To disseminate an Item as a package, give the command:

```
[dspace]/bin/packager -e user -d -i handle -t packager path
```

Where user is the e-mail address of the E-Person under whose authority this runs; handle is the Handle of the Item to disseminate; packager is the plugin name of the package disseminator to use; and path is the path to the file to create (or "-" to write to the standard output). This example writes an Item out as a METS package in the file "454.zip":

```
/dspace/bin/packager -e florey@mit.edu -d -i 1721.2/454 -t METS   454.zip
```

Base command

The import command will always start with


```
[dspace]/bin/import
```

And completed with a number of parameters

Parameters

Action

Exactly one action must be specified:

```
--add
```

Implies all items in the archive directory are added as new items in the DSpace repository

```
--replace
```

Implies all items in the archive directory having a corresponding line in the mapping file will be updated, while the items in the archive directory without a corresponding line in the mapping file are added as new items in the DSpace repository

```
--delete
```

Implies all items the DSpace repository corresponding to a line in the mapping file will be withdrawn from the repository

Input specification

```
--source=items_dir
```

Specifies the path to the archive directory which has been created before. The source is required when either the add or the replace action is used.

```
--collection=collectionID
```

Specifies the collection(s) to which the items should all be added. The value can either be the internal collection id, or the collection handle (e.g. 123456789/35). The collection handle is a more useful format as this can be determined easily for each collection based on the URL in the browser. The collection is required when either the add or the replace action is used.

```
--mapfile=mapfile
```

Specifies the file containing the mapping between records in the archive directory and DSpace items. The mapfile is always required.

```
--eperson=joe@user.com
```

Specifies the DSpace Eperson to be attached as the submitter for the imported items. DSpace requires a submitter for each item. The submitter must have the required permissions for executing the action. Usually a general administrator account will be used for this task. The eperson is always required.

```
--workflow
```

If this parameter is added, the standard collection workflow will be used for all imported items. This implies that the metadata editors configured for the specified collection will receive a task for each of the imported items. If this parameter is omitted, the imported will be archived immediately.

```
--test
```

If this parameter is added, a test import will be executed. This implies that the metadata will be read, but not actually imported in the repository. A test import can be executed to find errors in the archive directory or missing parameters before executing an actual import. If this parameter is omitted, the actual import will be executed.

`--template`

If this parameter is added, the collection item template will be applied. The items will be imported starting with the metadata present in the item template, and the metadata present in the metadata xml files will be added. If this parameter is omitted, the item template will be ignored and the imported items will only contain the metadata present in the metadata xml files.

`--resume`

If this parameter is added, the mapfile will be used to determine which items were previously added to the repository. The remaining items will be imported into the repository. This is especially useful for failed imports.

Exercises

2.1. Disseminating

Export an item containing files from your repository (e.g. the one you've uploaded in the itemimporter), using the METS exporter. Keep in mind that the METS exporter will create zip files, so using a filename with the zip extension is preferred.

Open the file which has been generated by the disseminator, and compare the contents to the item in your repository. Look for the file descriptors and find the characteristics offered per file.

2.2. Ingesting

Import the zip file you've exported before as a new item in your repository. Compare the now uploaded item to the item you've exported before.

3. SWORD

The contents of this section is based on "The DSpace Course - SWORD basics" by Lewis, Stuart and Yates, Chris[1] and the SWORD project website[2]. The complete SWORD specifications can be found at http://purl.org/net/sword. This document only provides a very short overview of SWORD.

Basics

SWORD (Simple Web-service Offering Repository Deposit) is a common repository deposit protocol. It has created a standardized way of depositing content into repositories and was implemented for the DSpace, EPrints, Fedora and Intralibrary repositories. SWORD is an extension of the Atom Publishing Protocol[3], which is a simple HTTP-based protocol for creating and updating web resources.

SWORD was designed to be able to address 3 major depositing scenarios:

Simultaneous multiple deposit - A researcher may wish to deposit an item to multiple repositories at once: eg. Local institutional repository, Funders repository, and a Subject repository

Deposit by lab equipment - Researcher may wish to configure modern hi-tech lab equipment to deposit their data straight into a repository.

One-click deposit - A researcher may wish to deposit to a repository directly from their word processor software. If a SWORD client was built into a word processor it could be used to do this.

---

[1] http://hdl.handle.net/2160/633

[2] http://www.swordapp.org

[3] http://bitworking.org/projects/atom/draft-ietf-atompub-protocol-04.html

Terminology

IRI - An Internationalized Resource Identifier. Before an IRI found in a document is used by HTTP, the IRI is first converted to a URI.

Resource - A network-accessible data object or service identified by an IRI.

relation (or "relation of") - Refers to the "`rel`" attribute value of an `atom:link` element.

Representation - An entity included with a request or response.

Collection - A Resource that contains a set of Member Resources. Collections are represented as Atom Feeds.

Member (or Member Resource) - A Resource whose IRI is listed in a Collection by an `atom:link` element with a relation of "edit" or "edit-media". The protocol defines two kinds of Members:
- *Entry Resource* - Members of a Collection that are represented as Atom Entry Documents.
- *Media Resource* - Members of a Collection that have representations other than Atom Entry Documents.

Media Link Entry - An Entry Resource that contains metadata about a Media Resource.

Workspace - A named group of Collections.

Service Document - A document that describes the location and capabilities of one or more Collections, grouped into Workspaces.

SWORD Profile of AtomPub - Protocol Operations

Example

```
POST /geography-collection HTTP/1.1
Host: www.myrepository.ac.uk
Content-Type: application/zip
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Length: nnn
Content-MD5: [md5-digest]
Content-Disposition: filename=myDSpaceMETSItem.zip
X-Packaging: http://purl.org/net/sword-types/mets/dspace
User-Agent: MyJavaClient/0.1 Restlet/2.0

HTTP/1.1 201 Created
Date: Mon, 18 August 2008 14:27:11 GMT
Content-Length: nnn
Content-Type: application/atom+xml; charset="utf-8"
Location: http://www.myrepository.ac.uk/geography-collection/atom/my_deposit.atom

<?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>
   <sword:userAgent>MyJavaClient/0.1 Restlet/2.0</sword:userAgent>
   <generator uri="http://www.myrepository.ac.uk/engine" version="1.0"/>
   <content type="application/zip"
      src="http://www.myrepository.ac.uk/geography-collection/deposit1.zip"/>
   <sword:packaging>http://purl.org/net/sword-types/mets/dspace</sword:packaging>
   <link rel="edit"
      href="http://www.myrepository.ac.uk/geography-collection/atom/my_deposit.atom" />
</entry>
```


Retrieving a Service Document

The client sends a `GET` request to the URI of the Service Document. The server responds with a Service Document enumerating a group of Collections and the capabilities of those Collections supported by the server. In addition, SWORD defines an additional HTTP header `X-On-Behalf-Of` used to specify the username of a target user on whose behalf a deposit is being made. When a server that supports mediated deposit receives an X-On-Behalf-Of header, the returned Service Document should identify only those collections to which the combination of mediated user and authenticated user might successfully deposit.

Listing Collections

Under the SWORD profile, implementations should provide representations in the form of Atom Feed Documents. Clients must not require a Collection Feed Document for deposit operation.

Creating a Resource

The client POSTs a representation of the Member to the URI of the Collection. If the Member Resource was created successfully, the server responds with a status code of 201 and a Location header that contains the IRI of the newly created Entry Resource. Media Resources could have also been created and their IRIs can be found through the Entry Resource.

Response:

```
201 Created
Location: [Member Entry URI]
[Optional Atom Entry document]
```

Editing a Resource

Once a Resource has been created and its Member URI is known, that URI can be used to retrieve, edit, and delete the Resource.

Retrieving a Resource - The client sends a `GET` request to the URI of a Member Resource to retrieve its representation. The server responds with the representation of the Member Resource.

Editing a Resource - The client sends a `PUT` request to store a representation of a Member Resource. If the request is successful, the server responds with a status code of 200.

Deleting a Resource - The client sends a `DELETE` request to the URI of a Member Resource. If the deletion is successful, the server responds with a status code of 200.

SWORD in DSpace

DSpace is built using a modular structure, each module adds a new piece of functionality. Sword is one of the DSpace modules and can be found in `[dspace-source]/dspace-sword`. Other modules include the JSP user interface, and the XML user interface (Manakin), and the OAI-PMH interface.

In order to use SWORD in your repository, simply deploy the compiled sword webapp which can be found at `[dspace]/webapps/sword` after a `maven clean package` has been executed. Hereafter, the sword deposit interface is available at http://localhost:8080/sword/deposit or a similar URL.

There are some configuration parameters specific to SWORD in DSpace:

- The value of `sword.mets-ingester.package-ingester` tells the system which named plugin for this interface should be used to ingest SWORD METS packages. The default value is `METS`.
- The value of `mets.submission.crosswalk.EPDCX` defines the metadata type EPDCX (EPrints DC XML) to be handled by the SWORD crosswalk configuration. The default value is `SWORD`.

- The value of `crosswalk.submission.SWORD.stylesheet` defines the stylesheet which will be used by the self-named XSLTIngestionCrosswalk class *(see OAI session for more information)* when asked to load the SWORD configuration (as specified above). This will use the specified stylesheet to crosswalk the incoming SWAP metadata to the DIM format for ingestion. The default configuration specifies `crosswalks/sword-swap-ingest.xsl` as the stylesheet.
- The base URL of the SWORD deposit can be configured with `sword.deposit.url` and defaults to `{dspace.url}/sword/deposit`.
- The metadata field in which to store the updated date for items deposited via SWORD can be configured using `sword.updated.field` which is configured by default to `dc.date.updated`.
- The metadata field in which to store the value of the slug header if it is supplied can be configured using `sword.slug.field` which is configured by default to `dc.identifier.slug`.

Deposits in DSpace

In a default configuration DSpace will accept certain packages for deposit. The packages must be a collection of files zipped up together with a manifest file encoded in METS (Metadata Encoding and Transmission Standard) with the metadata in SWAP (Scholarly Works Application Profile) format. New package types could be supported through the use of the standard DSpace packager functionality.

Exercise

Register an account at http://dspace.swordapp.org/jspui/register. Use http://client.swordapp.org/client/ with http://dspace.swordapp.org/sword/servicedocument selected and the username and password just created to retrieve the service document. Deposit in the "DSpace SWORD 1.3 Demo" collection using a METS package previously exported from DSpace using the package exporter *(see the import/export session)* or from http://www.ukoln.ac.uk/repositories/sword/example.zip.

Packaging format: http://purl.org/net/sword-types/METSDSpaceSIP

File Type (MIME type): application/zip

4. Solutions to Exercises

Exercise 1.1.

```
dspace/bin/import --add --eperson=youraccount@mail.com --collection=1
                  --source=[dspace]/archive_directory
                  --mapfile=[dspace]/mapfiles/mapfile
```

Exercise 1.2.

```
dspace/bin/import --resume --eperson=youraccount@mail.com --collection=1
                  --source=[dspace]/archive_directory
                  --mapfile=[dspace]/mapfiles/mapfile
```

Exercise 2.1.

```
packager -e youraccount@mail.com -d -i 123456789/4 -t METS [dspace]/mets.zip
```

Exercise 2.2.

```
packager -e youraccount@mail.com -c 123456789/2 -t METS [dspace]/mets.zip
```


TERMS OF USE

PLEASE READ THESE TERMS OF USE CAREFULLY BEFORE USING THESE COURSE MATERIALS. By using these course materials, you signify your assent to these terms of use. If you do not agree to these terms of use, please do not use these course materials.

RESTRICTIONS ON USE OF MATERIALS. These course materials are owned by @mire NV, Technologielaan 9, 3001 Heverlee (Belgium).

No components from these course materials owned, licensed or controlled by @mire NV may be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, except that you may download one copy of the materials on any single computer for your personal, non-commercial home use only, provided you keep intact all copyright and other proprietary notices.

Modification of the materials or use of the materials for any other purpose is a violation of @mire's copyright and other proprietary rights. The use of any such material on any other web site or networked computer environment is prohibited.

To request permission to reproduce materials,
call +32 2 888 29 56,
email info@atmire.com,
or write to @mire NV, Technologielaan 9, 3001 Heverlee, Belgium.

