SWORDv1 Server

SWORD (Simple Web-service Offering Repository Deposit) is a protocol that allows the remote deposit of items into repositories. DSpace implements the SWORD protocol via the 'sword' web application. The version of SWORD v1 currently supported by DSpace is 1.3. The specification and further information can be found at http://swordapp.org.

SWORD is based on the Atom Publish Protocol and allows service documents to be requested which describe the structure of the repository, and packages to be deposited.

Enabling SWORD Server

To enable DSpace's SWORD server, just make sure the `[dspace]/webapps/sword/` web application is available from your Servlet Container (usually Tomcat).

Configuring SWORD Server

These are the SWORD (v1) configurations. They may be edited directly or overridden in your local.cfg config (see Configuration Reference).

Configuration File:	**`[dspace]/config/modules/sword-server.cfg`**
Property:	`sword-server.mets-ingester.package-ingester`
Example Value:	`sword-server.mets-ingester.package-ingester = METS`
Informational Note:	The property key tell the SWORD METS implementation which package ingester to use to install deposited content. This should refer to one of the classes configured for: `plugin.named.org.dspace.content.packager.PackageIngester` 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.

Properties:	`mets.default.ingest.crosswalk.EPDCX`
	`mets.default.ingest.crosswalk.*`
	(NOTE: These configs are in the `dspace.cfg` file as they are used by many interfaces)

Example Value:	`mets.submission.crosswalk.EPDCX = EPDCX`
Informational Note:	Define the metadata types (METS "mdtype") which can be accepted/handled by SWORD during ingest of a package. Currently, EPDCX (EPrints DC XML) is the recommended default metadata format, but others are supported. An example of an EPDCX SWORD package can be found at [dspace-src]/dspace-sword/example/example.zip. Additional metadata types can be added to this list by just defining new configurations. For example, you can map a new "mdtype" MYFORMAT to a custom crosswalk named MYFORMAT: `mets.submission.crosswalk.MYFORMAT = MYFORMAT` You'd also want to map your new custom crosswalk to a stylesheet using the next configuration (crosswalk.submission.*.stylesheet).
Property:	`crosswalk.submission.EPDCX.stylesheet` (NOTE: This configuration is in the `dspace.cfg` file)
Example Value:	`crosswalk.submission.EPDCX.stylesheet = crosswalks/sword-swap-ingest.xsl`
Informational Note:	Define the stylesheet which will be used by the self-named XSLTIngestionCrosswalk class when asked to load the SWORD configuration (as specified in previous setting). This will use the specified stylesheet to crosswalk the incoming SWAP metadata to the DIM format for ingestion. Additional crosswalk types can be added to this list by just defining new configurations. For example, you can map a custom crosswalk named MYFORMAT to use a specific "my-crosswalk.xsl" stylesheet: `crosswalk.submission.MYFORMAT.stylesheet = crosswalks/my-crosswalk.xsl` Keep in mind, you'll need to also ensure MYFORMAT crosswalk is defined by the previous configuration (mets.submission.crosswalk.*).

Property:	`sword-server.deposit.url`
Example Value:	`sword-server.deposit.url = http://www.myu.ac.uk/sword/deposit`
Informational Note:	The base URL of the SWORD deposit. This is the URL from which DSpace will construct the deposit location URLs for collections. The default is `${dspace.baseUrl}/sword/deposit` (where `dspace.baseUrl` is defined in your `dspace.cfg` file). In the event that you are not deploying DSpace as the ROOT application in the servlet container, this will generate incorrect URLs, and you should override the functionality by specifying in full as shown in the example value.
Property:	`sword-server.servicedocument.url`
Example Value:	`sword-server.servicedocument.url = http://www.myu.ac.uk/sword/servicedocument`
Informational Note:	The base URL of the SWORD service document. This is the URL from which DSpace will construct the service document location URLs for the site, and for individual collections. The default is `${dspace.baseUrl}/sword/servicedocument` (where `dspace.baseUrl` is defined in your `dspace.cfg` file). In the event that you are not deploying DSpace as the ROOT application in the servlet container, this will generate incorrect URLs, and you should override the functionality by specifying in full as shown in the example value.
Property:	`sword-server.media-link.url`
Example Value:	`sword-server.media-link.url = http://www.myu.ac.uk/sword/media-link`
Informational Note:	The base URL of the SWORD media links. This is the URL which DSpace will use to construct the media link URLs for items which are deposited via sword. The default is `${dspace.baseUrl}/sword/media-link` (where `dspace.baseUrl` is defined in your `dspace.cfg` file). In the event that you are not deploying DSpace as the ROOT application in the servlet container, this will generate incorrect URLs, and you should override the functionality by specifying in full as shown in the example value.
Property:	`sword-server.generator.url`
Example Value:	`sword-server.generator.url = http://www.dspace.org/ns/sword/1.3.1`
Informational Note:	The URL which identifies the SWORD software which provides the sword interface. This is the URL which DSpace will use to fill out the atom:generator element of its atom documents. The default is: http://www.dspace.org/ns/sword/1.3.1 If you have modified your SWORD software, you should change this URI to identify your own version. If you are using the standard 'dspace-sword' module you will not, in general, need to change this setting.
Property:	`sword-server.updated.field`
Example Value:	`sword-server.updated.field = dc.date.updated`
Informational Note:	The metadata field in which to store the updated date for items deposited via SWORD.
Property:	`sword-server.slug.field`
Example Value:	`sword-server.slug.field = dc.identifier.slug`
Informational Note:	The metadata field in which to store the value of the slug header if it is supplied.
Properties:	`sword-server.accept-packaging.METSDSpaceSIP.identifier` `sword-server.accept-packaging.METSDSpaceSIP.q`
Example Value:	`sword-server.accept-packaging.METSDSpaceSIP.identifier = http://purl.org/net/sword-types/METSDSpaceSIP` `sword-server.accept-packaging.METSDSpaceSIP.q = 1.0`

Informational Note:	The accept packaging properties, along with their associated quality values where appropriate. This is a Global Setting; these will be used on all DSpace collections
Property:	`sword-server.accepts`
Example Value:	`sword-server.accepts = application/zip, foo/bar`
Informational Note:	A comma separated list of MIME types that SWORD will accept.
Properties:	`sword-server.accept-packaging.[handle].METSDSpaceSIP.identifier` `sword-server.accept-packaging.[handle].METSDSpaceSIP.q`
Example Value:	`sword-server.accept-packaging.[handle].METSDSpaceSIP.identifier = http://purl.org/net/sword-types/METSDSpaceSIP` `sword-server.accept-packaging.[handle].METSDSpaceSIP.q = 1.0`
Informational Note:	Collection Specific settings: these will be used on the collections with the given handles.
Property:	`sword-server.expose-items`
Example Value:	`sword-server.expose-items = false`
Informational Note:	Should the server offer up items in collections as sword deposit targets. This will be effected by placing a URI in the collection description which will list all the allowed items for the depositing user in that collection on request. **NOTE:** this will require an implementation of deposit onto items, which will not be forthcoming for a short while.
Property:	`sword-server.expose-communities`
Example Value:	`sword-server.expose-communities = false`
Informational Note:	Should the server offer as the default the list of all Communities to a Service Document request. If false, the server will offer the list of all collections, which is the default and recommended behavior at this stage. **NOTE:** a service document for Communities will not offer any viable deposit targets, and the client will need to request the list of Collections in the target before deposit can continue.
Property:	`sword-server.max-upload-size`
Example Value:	`sword-server.max-upload-size = 0`
Informational Note:	The maximum upload size of a package through the sword interface, in bytes. This will be the combined size of all the files, the metadata and any manifest data. It is NOT the same as the maximum size set for an individual file upload through the user interface. If not set, or set to 0, the sword service will default to no limit.
Property:	`sword-server.keep-original-package`
Example Value:	`sword-server.keep-original-package = true`
Informational Note:	Whether or not DSpace should store a copy of the original sword deposit package. **NOTE:** this will cause the deposit process to run slightly slower, and will accelerate the rate at which the repository consumes disk space. BUT, it will also mean that the deposited packages are recoverable in their original form. It is strongly recommended, therefore, to leave this option turned on. When set to "true", this requires that the configuration option `upload.temp.dir` (in `dspace.cfg`) is set to a valid location.
Property:	`sword-server.bundle.name`
Example Value:	`sword-server.bundle.name = SWORD`
Informational Note:	The bundle name that SWORD should store incoming packages under if sword.keep-original-package is set to true. The default is "SWORD" if not value is set
Properties:	`sword-server.keep-package-on-fail` `sword-server.failed-package.dir`
Example Value:	`sword-server.keep-package-on-fail=true` `sword-server.failed-package.dir=${dspace.dir}/upload`
Informational Note:	In the event of package ingest failure, provide an option to store the package on the file system. The default is false.

Property:	`sword-server.identify-version`
Example Value:	`sword-server.identify-version = true`
Informational Note:	Should the server identify the sword version in a deposit response. It is recommended to leave this unchanged.
Property:	`sword-server.on-behalf-of.enable`
Example Value:	`sword-server.on-behalf-of.enable = true`
Informational Note:	Should mediated deposit via sword be supported. If enabled, this will allow users to deposit content packages on behalf of other users.
Property:	`sword-server.restore-mode.enable`
Example Value:	`sword-server.restore-mode.enable = true`
Informational Note:	Should the sword server enable restore-mode when ingesting new packages. If this is enabled the item will be treated as a previously deleted item from the repository. If the item had previously been assigned a handle then that same handle will be restored to activity. If that item had not been previously assign a handle, then a new handle will be assigned.
Property:	`plugin.named.org.dspace.sword.SWORDingester`

Example Value:	`plugin.named.org.dspace.sword.SWORDIngester = \`
	`  org.dspace.sword.SWORDMETSIngester = http://purl.org/net/sword-types/METSDSpaceSIP \`
	`  org.dspace.sword.SimpleFileIngester = SimpleFileIngester`

Informational Note:	Configure the plugins to process incoming packages. The form of this configuration is as per the Plugin Manager's Named Plugin documentation: `plugin.named.[interface] = [implementation] = [package format identifier]` (see `dspace.cfg`). Package ingesters should implement the SWORDIngester interface, and will be loaded when a package of the format specified above in: `sword-server.accept-packaging.[package format].identifier = [package format identifier]` is received. In the event that this is a simple file deposit, with no package format, then the class named by "SimpleFileIngester" will be loaded and executed where appropriate. This case will only occur when a single file is being deposited into an existing DSpace Item.

Deposit to SWORD Server

If you'd like to deposit content to your repository via the installed SWORD Server, you'll need to select a SWORD Client to do so.

- A variety of SWORDv1 Clients (in various languages/tools) are available off of http://swordapp.org/sword-v1/
- The DSpace XMLUI also comes with an optional SWORDv1 Client which can be enabled to deposit content from one DSpace to another.
- An example SWORDv1 ZIP package is available in the DSpace Codebase at: https://github.com/DSpace/DSpace/tree/dspace-5_x/dspace-sword/example
- Finally, it's also possible to simply deposit a valid SWORD Zip package via common Linux commandline tools (e.g. curl). For example:

```
# Deposit a SWORD Zip package named "sword-package.zip" into a DSpace Collection (Handle 123456789/2) as
user "test@dspace.org"
# (Please note that you WILL need to obviously modify the Collection location, user/password and name of
the SWORD package)

curl -i --data-binary "@sword-package.zip" -H "Content-Disposition:filename=sword-package.zip" -H
"Content-Type:application/zip" -H "X-Packaging:http://purl.org/net/sword-types/METSDSpaceSIP" -u
test@dspace.org:[password] http://[dspace.url]/sword/deposit/123456789/2

# Template 'curl' command:
#curl -i --data-binary "@[zip-package-name]" -H "Content-Disposition:filename=[zip-package-name]" -H
"Content-Type:application/zip" -H "X-Packaging:http://purl.org/net/sword-types/METSDSpaceSIP" -u [user]:
[password] http://[dspace.url]/sword/deposit/[collection-handle]
```

DSpace 6 Demo - Service Document

- https://demo.dspace.org/sword/servicedocument
- https://demo.dspace.org/swordv2/servicedocument