SWORDv2 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 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 v2 Server

To enable DSpace's SWORD v2 server, just make sure the `[dspace]/webapps/swordv2/` web application is available from your Servlet Container (usually Tomcat).

Configuring SWORD v2 Server

These are the SWORD (v2) configurations. They may be edited directly or overridden in your local.cfg config (see Configuration Reference).

Configuration File:	**`[dspace]/config/modules/swordv2-server.cfg`**
Property:	`swordv2-server.url`
Example Value:	`swordv2-server.url = http://www.myu.ac.uk/swordv2`
Informational Note:	The base url of the SWORD 2.0 system. This defaults to `${dspace.baseUrl}/swordv2` (where `dspace.baseUrl` is defined in your `dspace.cfg` file).
Property:	`swordv2-server.collection.url`
Example Value:	`swordv2-server.collection.url = http://www.myu.ac.uk/swordv2/collection`
Informational Note:	The base URL of the SWORD collection. This is the URL from which DSpace will construct the deposit location URLs for collections. This defaults to `${dspace.baseUrl}/swordv2/collection` (where `dspace.baseUrl` is defined in your `dspace.cfg` file).
Property:	`swordv2-server.servicedocument.url`
Example Value:	`swordv2-server.servicedocument.url = http://www.myu.ac.uk/swordv2/servicedocument`

Informational Note:	The service document URL of the SWORD collection. 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. This defaults to `${dspace.baseUrl}/swordv2/servicedocument` (where `dspace.baseUrl` is defined in your `dspace.cfg` file).
Property:	`swordv2-server.accept-packaging.collection`

Example Value:	`swordv2-server.accept-packaging.collection.METSDSpaceSIP = http://purl.org/net/sword/package/METSDSpaceSIP`
	`swordv2-server.accept-packaging.collection.SimpleZip = http://purl.org/net/sword/package/SimpleZip`
	`swordv2-server.accept-packaging.collection.Binary = http://purl.org/net/sword/package/Binary`

Informational Note:	The accept packaging properties, along with their associated quality values where appropriate. Package format information • METSDSpaceSIP: zipfile containing mets.xml file describing the resources packed together with it in the root of the zipfile. • Binary: Binary resource that should be taken in as-is, not unpacked • SimpleZip: Zip file that should be unpacked and each file in the zip should be ingested separately. No metadata provided/ingested.
Property:	`swordv2-server.accept-packaging.item`

Example Value:	`swordv2-server.accept-packaging.item.METSDSpaceSIP = http://purl.org/net/sword/package/METSDSpaceSIP`
	`swordv2-server.accept-packaging.item.SimpleZip = http://purl.org/net/sword/package/SimpleZip`
	`swordv2-server.accept-packaging.item.Binary = http://purl.org/net/sword/package/Binary`

Informational Note:	The accept packaging properties for items. It is possible to configure this for specific collections by adding the handle of the collection to the setting, for example `swordv2-server.accept-packaging.collection.[handle].METSDSpaceSIP = http://purl.org/net/sword-types/METSDSpaceSIP` Package format information • METSDSpaceSIP: zipfile containing mets.xml file describing the resources packed together with it in the root of the zipfile. • Binary: Binary resource that should be taken in as-is, not unpacked • SimpleZip: Zip file that should be unpacked and each file in the zip should be ingested separately. No metadata provided/ingested.
Property:	`swordv2-server.accepts`
Example Value:	`swordv2-server.accepts = application/zip, image/jpeg`
Informational Note:	A comma-separated list of MIME types that SWORD will accept. To accept all mimetypes, the value can be set to "*/*"
Property:	`swordv2-server.expose-communities`
Example Value:	`swordv2-server.expose-communities = false`
Informational Note:	Whether or not the server should expose a list of all the communities to a service document request. As deposits can only be made into a collection, it is recommended to leave this set to false.
Property:	`swordv2-server.max-upload-size`
Example Value:	`swordv2-server.max-upload-size = 0`
Informational Note:	The maximum upload size of a package through the SWORD interface (measured in bytes). This will be the combined size of all the files, metadata, and manifest file in a package - this is different to the maximum size of a single bitstream. If this is set to 0, no maximum file size will be enforced.
Property:	`swordv2-server.keep-original-package`
Example Value:	`swordv2-server.keep-original-package = true`

Informational Note:	Should DSpace store a copy of the orignal SWORD deposit package? This will cause the deposit process to be slightly slower and for more disk to be used, however original files will be preserved. It is recommended to leave this option enabled.
Property:	`swordv2-server.bundle.name`
Example Value:	`swordv2-server.bundle.name = SWORD`
Informational Note:	The bundle name that SWORD should store incoming packages within if `swordv2-server.keep-original-package` is set to true.
Property:	`swordv2-server.bundle.deleted`
Example Value:	`swordv2-server.bundle.deleted = DELETED`
Informational Note:	The bundle name that SWORD should use to store deleted bitstreams if `swordv2-server.versions.keep` is set to true. This will be used in the case that individual files are updated or removed via SWORD. If the entire Media Resource (files in the ORIGINAL bundle) is removed this will be backed up in its entirety in a bundle of its own
Property:	`swordv2-server.keep-package-on-fail`
Example Value:	`swordv2-server.keep-package-on-fail = false`
Informational Note:	In the event of package ingest failure, provide an option to store the package on the file system. The default is false. The location can be set using the `swordv2-server.failed-package-dir` setting.
Property:	`swordv2-server.failed-package-dir`
Example Value:	`swordv2-server.failed-package-dir = /dspace/upload`
Informational Note:	If `swordv2-server.keep-package-on-fail` is set to true, this is the location where the package would be stored.
Property:	`swordv2-server.on-behalf-of.enable`
Example Value:	`swordv2-server.on-behalf-of.enable = true`
Informational Note:	Should DSpace accept mediated deposits? See the SWORD specification for a detailed explanation of deposit On-Behalf-Of another user.

Property:	`swordv2-server.on-behalf-of.update.mediators`
Example Value:	`swordv2-server.on-behalf-of.update.mediators = admin@mydspace.edu, mediator@mydspace.edu`
Informational Note:	Which user accounts are allowed to do updates on items which already exist in DSpace, on-behalf-of other users? If this is left blank, or omitted, then all accounts can mediate updates to items, which could be a security risk, as there is no implicit checking that the authenticated user is a "legitimate" mediator
Property:	`swordv2-server.verbose-description.receipt.enable`
Example Value:	`swordv2-server.verbose-description.receipt.enable = false`
Informational Note:	Should the deposit receipt include a verbose description of the deposit? For use by developers - recommend to set to "false" for production systems
Property:	`swordv2-server.verbose-description.error.enable`
Example Value:	`swordv2-server.verbose-description.error.enable = true`
Informational Note:	should the error document include a verbose description of the error? For use by developers, although you may also wish to leave this set to "true" for production systems
Property:	`swordv2-server.error.alternate.url`
Example Value:	`swordv2-server.error.alternate.url = http://mydspace.edu/xmlui/contact`
Informational Note:	The error document can contain an alternate url, which the client can use to follow up any issues. For example, this could point to the Contact-Us page on the XMLUI
Property:	`swordv2-server.error.alternate.content-type`
Example Value:	`swordv2-server.error.alternate.content-type = text/html`
Informational Note:	The `swordv2-server.error.alternate.url` may have an associated content type, such as `text/html` if it points to a web page. This is used to indicate to the client what content type it can expect if it follows that url.
Property:	`swordv2-server.generator.url`
Example Value:	`swordv2-server.generator.url = http://www.dspace.org/ns/sword/2.0/`

Informational Note:	The URL which identifies DSpace as the software that is providing the SWORD interface.
Property:	`swordv2-server.generator.version`
Example Value:	`swordv2-server.generator.version = 2.0`
Informational Note:	The version of the SWORD interface.
Property:	`swordv2-server.auth-type`
Example Value:	`swordv2-server.auth-type = Basic`
Informational Note:	Which form of authentication to use. Normally this is set to `Basic` in order to use HTTP Basic.
Property:	`swordv2-server.upload.tempdir`
Example Value:	`swordv2-server.upload.tempd = /dspace/upload`
Informational Note:	The location where uploaded files and packages are stored while being processed.
Property:	`swordv2-server.updated.field`
Example Value:	`swordv2-server.updated.field = dc.date.updated`
Informational Note:	The metadata field in which to store the updated date for items deposited via SWORD.
Property:	`swordv2-server.slug.field`
Example Value:	`swordv2-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.
Property:	`swordv2-server.author.field`
Example Value:	`swordv2-server.author.field = dc.contributor.author`

Informational Note:	The metadata field in which to store the value of the atom entry author if it supplied.
Property:	`swordv2-server.title.field`
Example Value:	`swordv2-server.title.field = dc.title`
Informational Note:	The metadata field in which to store the value of the atom entry title if it supplied.
Property:	`swordv2-server.disseminate-packaging`
Example Value:	`swordv2-server.disseminate-packaging.METSDSpaceSIP = http://purl.org/net/sword/package/METSDSpaceSIP` `swordv2-server.disseminate-packaging.SimpleZip = http://purl.org/net/sword/package/SimpleZip`
Informational Note:	Supported packaging formats for the dissemination of packages.
Property:	`swordv2-server.statement.bundles`
Example Value:	`swordv2-server.statement.bundles = ORIGINAL, SWORD, LICENSE`
Informational Note:	Which bundles should the Statement include in its list of aggregated resources? The Statement will automatically mark any bitstreams which are in the bundle identified by the `${bundle.name}` property, provided that bundle is also listed here (i.e. if you want Original Deposits to be listed in the Statement then you should add the SWORD bundle to this list)
Property:	`plugin.single.org.dspace.sword2.WorkflowManager`
Example Value:	`plugin.single.org.dspace.sword2.WorkflowManager = org.dspace.sword2.WorkflowManagerDefault`
Informational Note:	Which workflow manager to use.
Property:	`swordv2-server.workflowmanagerdefault.always-update-metadata`
Example Value	`swordv2-server.workflowmanagerdefault.always-update-metadata = true`
Informational Note	Should the WorkflowManagerDefault plugin allow updates to the item's metadata to take place on items which are in states other than the workspace (e.g. in the workflow, archive, or withdrawn) ?

Property:	`swordv2-server.workflowmanagerdefault.file-replace.enable`
Example Value	`swordv2-server.workflowmanagerdefault.file-replace.enable = false`
Informational Note	Should the server allow PUT to individual files? If this is enabled, then DSpace may be used with the DepositMO SWORD extensions, BUT the caveat is that DSpace does not formally support Bitstream replace, so this is equivalent to a DELETE and then a POST, which violates the RESTfulness of the server. The resulting file DOES NOT have the same identifier as the file it was replacing. As such it is STRONGLY RECOMMENDED to leave this option turned off unless working explicitly with DepositMO enabled client environments
Property:	`swordv2-server.mets-ingester.package-ingester`
Example Value:	`swordv2-server.mets-ingester.package-ingester = METS`
Informational Note:	Which package ingester to use for METS packages.
Property:	`swordv2-server.restore-mode.enable`
Example Value:	`swordv2-server.restore-mode.enable = false`
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 has previously been assigned a handle then that same handle will be restored to activity.
Property:	`swordv2-server.simpledc.*`
Example Value:	`swordv2-server.simpledc.abstract = dc.description.abstractsimpledc.date = dc.datesimpledc.rights = dc.rights`
Informational Note:	Configuration of metadata field mapping used by the SimpleDCEntryIngester, SimpleDCEntryDisseminator and FeedContentDisseminator
Property:	`swordv2-server.atom.*`
Example Value	`swordv2-server.atom.author = dc.contributor.author`
Informational Note:	Configuration of metadata field mapping used by the SimpleDCEntryIngester, SimpleDCEntryDisseminator and FeedContentDisseminator
Property:	`swordv2-server.metadata.replaceable`

Example Value	`swordv2-server.metadata.replaceable = dc.description.abstract, dc.rights, dc.title.alternative, dc.identifier.citation`
Informational Note	Used by SimpleDCEntryIngester: Which metadata fields can be replaced during a PUT to the Item of an Atom Entry document? Fields listed here are the ones which will be removed when a new PUT comes through (irrespective of whether there is a new incoming value to replace them)
Property:	`swordv2-server.multipart.entry-first`
Example Value:	`swordv2-server.multipart.entry-first = false`
Informational Note:	The order of precedence for importing multipart content. If this is set to `true` then metadata in the package will override metadata in the atom entry, otherwise the metadata in the atom entry will override that from the package.
Property:	`swordv2-server.workflow.notify`
Example Value:	`swordv2-server.workflow.notify = true`
Informational Note:	If the workflow gets started (the collection being deposited into has a workflow configured), should a notification get sent?
Property:	`swordv2-server.versions.keep`
Example Value:	`swordv2-server.versions.keep = true`
Informational Note:	When content is replaced, should the old version be kept? This creates a copy of the ORIGINAL bundle with the name V_YYYY-MM-DD.X where YYYY-MM-DD is the date the copy was created, and X is an integer from 0 upwards.
Property:	`swordv2-server.state.*`

Example Value:	swordv2-server.state.workspace.uri = http://localhost:8080/xmlui/state/inprogress swordv2-server.state.workspace.description = The item is in the user workspace swordv2-server.state.workflow.uri = http://localhost:8080/xmlui/state/inreview swordv2-server.state.workflow.description = The item is undergoing review prior to acceptance in the archive
Informational Note:	Pairs of states (URI and description) than items can be in. Typical states are `workspace`, `workflow`, `archive`, and `withdrawn`.
Property:	`swordv2-server.workspace.url-template`
Example Value	`swordv2-server.workspace.url-template = http://myspace.edu/xmlui/submit?workspaceID=#wsid#`
Informational Note	URL template for links to items in the workspace (items in the archive will use the handle). The `#wsid#` url parameter will be replaced with the workspace id of the item. The example above shows how to construct this URL for XMLUI.

Other configuration options exist that define the mapping between mime types, ingesters, and disseminators. A typical configuration looks like this:

```
plugin.named.org.dspace.sword2.SwordContentIngester = \
  org.dspace.sword2.SimpleZipContentIngester = http://purl.org/net/sword
/package/SimpleZip, \
  org.dspace.sword2.SwordMETSIngester = http://purl.org/net/sword/package
/METSDSpaceSIP, \
  org.dspace.sword2.BinaryContentIngester = http://purl.org/net/sword
/package/Binary

plugin.single.org.dspace.sword2.SwordEntryIngester = \
  org.dspace.sword2.SimpleDCEntryIngester

plugin.single.org.dspace.sword2.SwordEntryDisseminator = \
  org.dspace.sword2.SimpleDCEntryDisseminator

# note that we replace ";" with "_" as ";" is not permitted in the
PluginManager names
plugin.named.org.dspace.sword2.SwordContentDisseminator = \
  org.dspace.sword2.SimpleZipContentDisseminator = http://purl.org/net
/sword/package/SimpleZip, \
  org.dspace.sword2.FeedContentDisseminator = application/atom+xml, \
  org.dspace.sword2.FeedContentDisseminator = application
/atom+xml_type_feed

# note that we replace ";" with "_" as ";" is not permitted in the
PluginManager names
plugin.named.org.dspace.sword2.SwordStatementDisseminator = \
  org.dspace.sword2.AtomStatementDisseminator = atom, \
  org.dspace.sword2.OreStatementDisseminator = rdf, \
  org.dspace.sword2.AtomStatementDisseminator = application
/atom+xml_type_feed, \
  org.dspace.sword2.OreStatementDisseminator = application/rdf+xml
```

Deposit to SWORDv2 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.

- There are currently no SWORDv2 Clients available at http://swordapp.org/sword-v2/
- It's possible to simply deposit a valid SWORDv2 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:
attachment; filename=sword-package.zip" -H "Content-Type:application
/zip" -H "Packaging:http://purl.org/net/sword/package/METSDSpaceSIP" -
u test@dspace.org:[password] -X POST http://[dspace.url]/swordv2
/collection/123456789/2

# Template 'curl' command:
#curl -i --data-binary "@[zip-package-name]" -H "Content-Disposition:
attachment; filename=[zip-package-name]" -H "Content-Type:application
/zip" -H "Packaging:http://purl.org/net/sword/package/METSDSpaceSIP" -
u [user]:[password] -X POST http://[dspace.url]/swordv2/collection/
[collection-handle]
```

Other example SWORDv2 commands

```
# Example of retrieving Item information via "edit-media" path in ATOM
format (can be run on any item within DSpace, but requires authentication)
# NOTE: Accept header is required, and must be a format supported by a
SwordContentDisseminator plugin (see configuration above)
curl -i -H "Accept:application/atom+xml" -u test@dspace.org:[password] -X
GET http://[dspace.url]/swordv2/edit-media/[internal-item-identifier]
```

Troubleshooting

Missing expression of encoding in XML header

If your SWORD Deposit requests are unsuccessful, please check that the XML in your initial metadata deposit correctly specifies the encoding.

If you use:

```
<?xml version="1.0"?>
```

DSpace will default to UTF-32.

So to successfully deposit an XML in UTF-8, make sure you use:

```
<?xml version="1.0" encoding="utf-8" ?>
```