

Importing and Exporting Items via Simple Archive Format (SAF)

- 1 [Item Importer and Exporter](#)
 - 1.1 [DSpace Simple Archive Format](#)
 - 1.1.1 [dublin_core.xml or metadata_\[prefix\].xml](#)
 - 1.1.2 [contents file](#)
 - 1.1.3 [relationships file](#)
 - 1.2 [Configuring metadata_\[prefix\].xml for a Different Schema](#)
 - 1.3 [Importing Items](#)
 - 1.3.1 [Adding Items to a Collection from a directory](#)
 - 1.3.2 [Adding Items to a Collection from a zipfile](#)
 - 1.3.3 [Replacing Items in a Collection](#)
 - 1.3.4 [Deleting or Unimporting Items in a Collection](#)
 - 1.3.5 [Other Options](#)
 - 1.3.6 [UI Batch Import](#)
 - 1.4 [Exporting Items](#)
 - 1.4.1 [UI Batch Export](#)

Item Importer and Exporter

DSpace has a set of command line tools for importing and exporting items in batches, using the DSpace Simple Archive Format (SAF). Apart from the offered functionality, these tools serve as an example for users who aim to implement their own item importer.

DSpace Simple Archive Format

The basic concept behind the DSpace's Simple Archive Format (SAF) is to create an archive, which is a directory containing one subdirectory per item. Each item directory contains a file for the item's descriptive metadata, and the files that make up the item.

```
archive_directory/
  item_000/
    dublin_core.xml      -- qualified Dublin Core metadata for metadata fields belonging to the 'dc'
schema.
    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.
      collections        -- (Optional) text file that contains the handles of
the collections the item will belong to. Each handle in a row.
      collection.        -- Collection in first line will be the owning
collection.
      handle             -- contains the handle assigned/to be assigned to
this resource
      relationships      -- (Optional) If importing Entities, you can specify one or more relationships
to create on import
      file_1.doc         -- files to be added as bitstreams to the item.
      file_2.pdf
  item_001/
    dublin_core.xml
    contents
    file_1.png
    ...
```

dublin_core.xml or metadata_[prefix].xml

The dublin_core.xml or metadata_[prefix].xml file has the following format, where each metadata element has its 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


```
<dublin_core>
  <dcvalue element="title" qualifier="none">A Tale of Two Cities</dcvalue>
  <dcvalue element="date" qualifier="issued">1990</dcvalue>
  <dcvalue element="title" qualifier="alternative" language="fr">J'aime les Printemps</dcvalue>
</dublin_core>
```

(Note the optional language tag attribute which notifies the system that the optional title is in French.)

When providing urls as values for fields that contain the ampersand (&) symbol, the ampersands in these urls have to be encoded as **&**;

Every metadata field used, must be registered via the metadata registry of the DSpace instance first. See [Metadata and Bitstream Format Registries](#).

Recommended Metadata

 It is recommended to minimally provide "dc.title" and, where applicable, "dc.date.issued". Obviously you can (and should) provide much more detailed metadata about the Item. For more information see: [Metadata Recommendations](#).

contents file

The `contents` file is a plain text document that simply enumerates, one file per line, the bitstream file names. See the following example:

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

Please notice that the `license` is optional, and if you wish to have one included, you can place the file in the `.../item_001/` directory, for example.

The bitstream name may *optionally* be followed by any of the following:

- `\tbundle:BUNDLENAME`
- `\tpermissions:PERMISSIONS`
- `\tdescription:DESCRIPTION`
- `\tprimary:true`

Where `\t` is the Tab character (not a literal `"\t"` string)

'BUNDLENAME' is the name of the bundle to which the bitstream should be added. Without specifying the bundle, items will go into the default bundle, ORIGINAL.


'PERMISSIONS' is text with the following format: `-[r|w] 'group name'`

- `-r` = read permissions for this group
- `-w` = write permissions for this group

'DESCRIPTION' is text of the files description.

Primary is used to specify the primary bitstream.

Supported in 7.2 or above for 'import' only

 The IIIF metadata feature was added in 7.2 and is only supported on *import* ('add' mode) of an SAF package.

For IIIF enabled items, the bitstream name may optionally be followed by any of the following:

- `\tiiif-label:IIIFLABEL`
- `\tiiif-toc:IIIFTOC`
- `\tiiif-width:IIIFWIDTH`
- `\tiiif-height:IIIFHEIGHT`

Where:

'IIIFLABEL' is the label that will be used for the image in the viewer.


'IIIFTOC' is the label that will be used for a table of contents entry in the viewer.

'IIIFWIDTH' is the image width that will be used for the IIIF canvas.

'IIIFHEIGHT' is the image height that will be used for the IIIF canvas.

relationships file

Supported in 7.1 or above for 'import' only.

 This feature was added in 7.1. Currently the 'relationships' file is only supported on *import* ('add' mode) of an SAF package. See note at bottom of this section about using the `"metadata_relation.xml"` if you wish to export & update relationships.

The optional `relationships` file enumerates the relationships of this Entity to other Entities (either already in the system, or also specified in your SAF import batch). This allows entities to be linked to new or existing entities during import. Entities can be linked to other entities in this import set by referring to their import subfolder name. Because relationships can only be created for Entities, it can only be used when importing [Configurable Entities](#).

Each line in the file contains a relationship type key and an item identifier in the following format:

```
relation.<relation_key> <handle|uuid|folderName:import_item_folder|schema.element[.qualifier]:value>
```

The `import_item_folder` should refer the folder name of another item in this import batch. Example:

```
relation.isAuthorOfPublication 5dace143-1238-4b4f-affb-ed559f9254bb
relation.isAuthorOfPublication 123456789/1123
relation.isOrgUnitOfPublication folderName:item_001
relation.isProjectOfPublication project.identifier.id:123
relation.isProjectOfPublication project.identifier.name:A Name with Spaces
```

During initial import, new items are stored in a map keyed by the item folder name. Once the initial import is complete, a second pass checks for a 'relationships' manifest file in each folder and creates a relationship of the specified type to the specified item.

Don't forget Entities require a "dspace.entity.type" metadata field



Remember, if you are creating *new* Entities via an SAF package, those Entities **MUST** specify a "dspace.entity.type" metadata field. Because this metadata field is in the "dspace" schema, it **MUST** be specified in a "metadata_dspace.xml", similar to:

metadata_dspace.xml

```
<dublin_core schema="dspace">
  <dcvalue element="entity" qualifier="type">Publication</dcvalue>
</dublin_core>
```

Relationships to existing Entities can also be created via `metadata_relation.xml`



If you already know the UUID of an existing Entity that you want to relate to, you can also create/update the "metadata_relation.xml" file to add/update the relationship, similar to:

metadata_relation.xml

```
<dublin_core schema="relation">
  <dcvalue element="isAuthorOfPublication">5dace143-1238-4b4f-affb-ed559f9254bb</dcvalue>
</dublin_core>
```

The "relationships" file is primarily for creating relationships between Entities in the *same import batch*. Of course, you can also choose to use the "relationships" file to create new relationships to existing Entities instead of creating/updating the "metadata_relation.xml" file. The main advantage of the "metadata_relation.xml" file is that it is used both on export and import, while the "relationships" file is only used on import at this time.

Configuring `metadata_[prefix].xml` for a Different Schema

It is possible to use other Schema such as EAD, VRA Core, etc. Make sure you have defined the new schema in the DSpace Metadata Schema Registry.

1. Create a separate file for the other schema named `metadata_[prefix].xml`, where the `[prefix]` is replaced with the schema's prefix.
2. Inside the xml file use the same Dublin Core *syntax*, but on the `<dublin_core>` element include the attribute `schema=[prefix]`.
3. Here is an example for ETD metadata, which would be in the file `metadata_etd.xml`:

```
<dublin_core schema="etd">
  <dcvalue element="degree" qualifier="department">Computer Science</dcvalue>
  <dcvalue element="degree" qualifier="level">Masters</dcvalue>
  <dcvalue element="degree" qualifier="grantor">Michigan Institute of Technology</dcvalue>
</dublin_core>
```

Importing Items

Before running the item importer over items previously exported from a DSpace instance, please first refer to Transferring Items Between DSpace Instances.

Command used:	<code>[dspace]/bin/dspace import</code>
---------------	---

Java class:	org.dspace.app.itemimport.ItemImport
Arguments short and (long) forms:	Description
-a or --add	Add items to DSpace ‡
-r or --replace	Replace items listed in mapfile ‡
-d or --delete	Delete items listed in mapfile ‡
-s or --source	Source of the items (directory)
-c or --collection	Destination Collection by its Handle or database ID
-m or --mapfile	Where the mapfile for items can be found (name and directory)
-e or --eperson	Email of eperson doing the importing
-w or --workflow	Send submission through collection's workflow
-n or --notify	Kicks off the email alerting of the item(s) has(have) been imported
-v or --validate	Test run, do not actually import items
-p or --template	Apply the collection template
-R or --resume	Resume a failed import (Used on Add only)
-h or --help	Command help
-z or --zip	Name of zipfile

‡ These are mutually exclusive.

The item importer is able to batch import unlimited numbers of items for a particular collection using a very simple CLI command and 'arguments'.

Adding Items to a Collection from a directory

To add items to a collection, you gather the following information:

- eperson
 - Collection ID (either Handle (e.g. 123456789/14) or UUID)
 - Source directory where the items reside
 - Mapfile. Since you don't have one, you need to determine where it will be (e.g. /Import/Col_14/mapfile)
- At the command line:

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

or by using the short form:

```
[dspace]/bin/dspace import -a -e joe@user.com -c CollectionID -s items_dir -m mapfile
```

The above command would cycle through the archive directory's items, import them, and then generate a map file which stores the mapping of item directories to item handles. **SAVE THIS MAP FILE.** You can use it for replacing or deleting (unimporting) the mapped items.

Testing. You can add --validate (or -v) to the command to simulate the entire import process without actually doing the import. This is extremely useful for verifying your import files before doing the actual import.

Adding Items to a Collection from a zipfile

To add items to a collection, you gather the following information:

- eperson
 - Collection ID (either Handle (e.g. 123456789/14) or Database ID (e.g. 2))
 - Source directory where your zipfile containing the items resides
 - Zipfile
 - Mapfile. Since you don't have one, you need to determine where it will be (e.g. /Import/Col_14/mapfile)
- At the command line:

```
[dspace]/bin/dspace import --add --eperson=joe@user.com --collection=CollectionID --source=zipfile_dir --zip=filename.zip --mapfile=mapfile
```

or by using the short form:

```
[dSPACE]/bin/dSPACE import -a -e joe@user.com -c CollectionID -s zipfile_dir -z filename.zip -m mapfile
```

The above command would unpack the zipfile, cycle through the archive directory's items, import them, and then generate a map file which stores the mapping of item directories to item handles. **SAVE THIS MAP FILE.** You can use it for replacing or deleting (unimporting) the mapped items.

Testing. You can add `--validate` (or `-v`) to the command to simulate the entire import process without actually doing the import. This is extremely useful for verifying your import files before doing the actual import.

Replacing Items in a Collection

Replacing existing items is relatively easy. Remember that mapfile you saved above? Now you will use it. The command (in short form):

```
[dSPACE]/bin/dSPACE import -r -e joe@user.com -c collectionID -s items_dir -m mapfile
```

Long form:

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

If you wish to replace content using a Zipfile, that's also possible. The command is similar. But, in this situation `-s` refers to the directory of the zip file, and `-z` gives the name of the zipfile:

```
[dSPACE]/bin/dSPACE import -r -e joe@user.com -c collectionID -s zipfile_dir -z filename.zip -m mapfile
```

Deleting or Unimporting Items in a Collection

You are able to unimport or delete items provided you have the mapfile. Remember that mapfile you saved above? The command is (in short form):

```
[dSPACE]/bin/dSPACE import -e joe@user.com -d -m mapfile
```

In long form:

```
[dSPACE]/bin/dSPACE import --eperson=joe@user.com --delete --mapfile mapfile
```

Other Options

- **Workflow.** The importer usually bypasses any workflow assigned to a collection. But add the `--workflow (-w)` argument will route the imported items through the workflow system.
- **Templates.** If you have templates that have constant data and you wish to apply that data during batch importing, add the `--template (-p)` argument.
- **Resume.** If, during importing, you have an error and the import is aborted, you can use the `--resume (-R)` flag to resume the import where you left off after you fix the error.
- **Specifying the owning collection on a per-item basis from the command line administration tool**

If you omit the `-c` flag, which is otherwise mandatory, the ItemImporter searches for a file named "collections" in each item directory. This file should contain a list of collections, one per line, specified either by their handle, or by their internal db id. The ItemImporter then will put the item in each of the specified collections. The owning collection is the collection specified in the first line of the collections file.

If both the `-c` flag is specified and the collections file exists in the item directory, the ItemImporter will ignore the collections file and will put the item in the collection specified on the command line.

Since the collections file can differ between item directories, this gives you more fine-grained control of the process of batch adding items to collections.

UI Batch Import

Available in DSpace 7.4 and above.

Batch import can also take place via the Administrator's UI. The steps to follow are:

A. Prepare the data

1. Items, i.e. the metadata and their bitstreams, must be in the Simple Archive Format described earlier in this chapter. Thus, for each item there must be a separate directory that contains the corresponding files of the specific item.
2. Moreover, in each item directory, there can be another file that describes the collection or the collections that this item will be added to. The name of this file must be "collections" and it is optional. It has the following format:

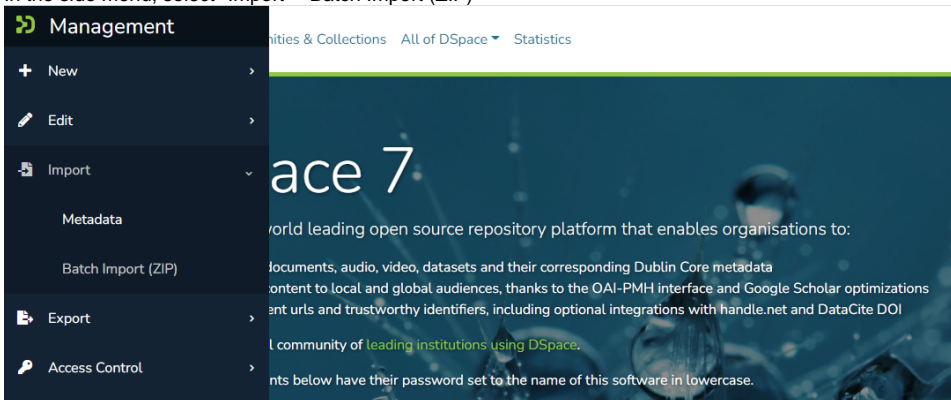
	Messages Properties	Main.jsp	Data
1	123456789/1		
2	123456789/5		
3	123456789/6		
4	123456789/9		
5			

Each line contains the handle of the collection. The collection in the first line is the owning collection while the rest are the other collections that the item should belong to.

3. Compress the item directories into a ZIP file. Please note that you need to zip the actual item directories and not just the directory that contains the item directories. Thus, the final zip file must directly contain the item directories.

B. Import the items via the UI

1. Login as an Administrator.
2. In the side menu, select "Import" "Batch Import (ZIP)"



3. From the "Import Batch" page:
 - a. Select the Collection you are importing into.
 - b. Drag & drop the ZIP file into the drop box (or browse to it on your filesystem).
 - c. Choose whether you want to "Validate Only" or not.
 - i. When selected, DSpace will **test** the batch import process, but no content will be batch imported. This allows you to validate the results of the import process before doing the import.
 - ii. When deselected, DSpace will do the batch import.

Import Batch

Select the Collection to import into. Then, drop or browse to a Simple Archive Format (SAF) zip file that includes the Items to import

Select Collection

☒ Validate Only
When selected, the uploaded ZIP will be validated. You will receive a report of detected changes, but no changes will be saved.

Drop a batch ZIP to import, or [browse](#)

Back

Proceed

4. Clicking "Proceed" will start the Batch Import. This creates a new "Process" which begins the upload of the batch. *Depending on the size of the batch, this process may take some time to complete.* You can refresh the page to see the current status, or go back to the list of processes ("Processes" menu in sidebar) to check on its status. Once the process is COMPLETED, you will see a log of the results and a mapfile (which can be used to make later updates).
5. All prior imports will be listed in the "Processes" menu, until their corresponding process entry is deleted. Once you are satisfied with the import and have no need to see the logs or mapfile, you may wish to delete that process entry in order to free up storage space (as your uploaded ZIP will be retained in DSpace until the process is deleted). A "process-cleaner" script can also be started from the "Processes" page which can be used to bulk delete old processes.

It is also possible to start an "import" directly from the "Processes" menu. This allows you to specify additional options/flags which are normally only available to the command-line "import" tool (see documentation above).

Exporting Items

The item exporter can export a single item or a collection of items, and creates a DSpace simple archive in [the aforementioned format](#) for each exported item. The items are exported in a sequential order in which they are retrieved from the database. As a consequence, the sequence numbers of the item subdirectories (item_000, item_001) are not related to DSpace handle or item IDs.

Command used:	[dspace]/bin/dspace export
Java class:	org.dspace.app.itemexport.ItemExport
Arguments short and (long) forms:	Description
-t or --type	Type of export. <i>COLLECTION</i> will inform the program you want the whole collection. <i>ITEM</i> will be only the specific item. (You will actually key in the keywords in all caps. See examples below.)
-i or --id	The ID or Handle of the Collection or Item to export.
-d or --dest	The destination path where you want the file of items to be placed.
-n or --number	Sequence number to begin with. Whatever number you give, this will be the name of the first directory created for your export. The layout of the export directory is the same as the layout used for import.
-m or --migrate	Export the item/collection for migration. This will remove the handle and any other metadata that will be re-created in the new instance of DSpace.
-x or --exclude-bitstreams	Do not export bitstreams. See the usage scenario below.
-h or --help	Brief Help.

Exporting a Collection

The CLI command to export the items of a collection:

```
[dspace]/bin/dspace export --type=COLLECTION --id=collectionID_or_handle --dest=/path/to/destination --number=seq_num
```

Short form:

```
[dspace]/bin/dspace export -t COLLECTION -i collectionID_or_handle -d /path/to/destination -n seq_num
```

The keyword *COLLECTION* means that you intend to export an entire collection. The ID can either be the database ID or the handle. The exporter will begin numbering the simple archives with the sequence number that you supply.

Exporting a Single Item

To export a single item use the keyword *ITEM* and give the item ID as an argument:

```
[dspace]/bin/dspace export --type=ITEM --id=itemID_or_handle --dest=/path/to/destination --number=seq_num
```

Short form:

```
[dspace]/bin/dspace export -t ITEM -i itemID_or_handle -d /path/to/destination -n seq_num
```

Each exported item will have an additional file in its directory, named "handle". This will contain the handle that was assigned to the item, and this file will be read by the importer so that items exported and then imported to another machine will retain the item's original handle.

The -m Argument

Using the -m argument will export the item/collection and also perform the migration step. It will perform the same process that the next section [Exchanging Content Between Repositories](#) performs. We recommend that section to be read in conjunction with this flag being used.

The -x Argument

Using the -x argument will do the standard export except for the bitstreams which will not be exported. If you have full SAF without bitstreams and you have the bitstreams archive (which might have been imported into DSpace earlier) somewhere near, you could [symlink](#) original archive files into SAF directories and have an exported collection which almost doesn't occupy any space but otherwise is identical to the exported collection (i.e. could be imported into DSpace). In case of huge collections -x mode might be substantially faster than full export.

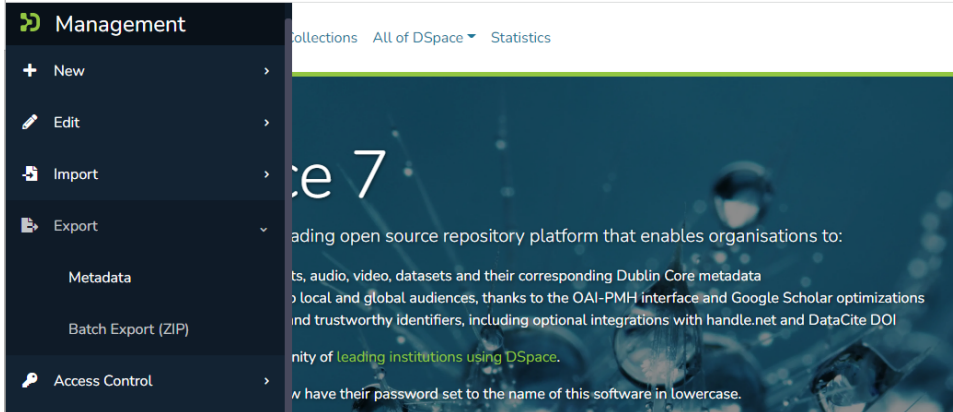
UI Batch Export

Available in DSpace 7.4 and above.

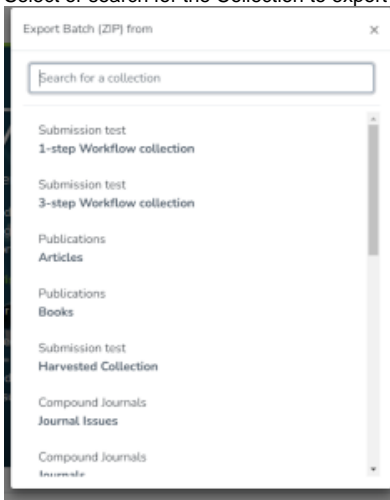
Batch export can also take place via the Administrator's UI. The default file size upload limit is 512MB, and is being configured in the [spring boot application.properties](#) file.

The steps to follow are:

1. Login as an Administrator.
2. In the side menu, select "Import" "Batch Export (ZIP)"



3. Select or search for the Collection to export from:



4. Clicking "Export" will start the Batch Export. This creates a new "Process" which begins export process. *Depending on the size of the export, this process may take some time to complete.* You can refresh to page to see the current status, or go back to the list of processes ("Processes" menu in sidebar) to check on its status. Once the process is COMPLETED, you will see a log of the results and an exported ZIP file which you can download for the results.
5. All prior exports will be listed in the "Processes" menu, until their corresponding process entry is deleted. Once you are satisfied with the export and have downloaded the ZIP, you may wish to delete that process entry in order to free up storage space (as your exported ZIP will be retained in DSpace until the process is deleted). A "process-cleaner" script can also be started from the "Processes" page which can be used to bulk delete old processes.

It is also possible to start an "export" directly from the "Processes" menu. This allows you to specify additional options/flags which are normally only available from the command-line "export" tool (see documentation above). It also allows you to export a single Item.