

# Import/export format

This section describes the import/export format. This is the format produced by the export tool, and expected by the import tool.

The format consists of a directory structure containing files (XML files and binary data files). This directory structure can optionally be zipped, though this is not required. Both the import and export tool support zipped or expanded structures.

## Overview

```
+ root
  + info
    + meta.xml
    + namespaces.xml
    + variants.xml
    + schema.xml
    + retired.xml
    + collections.xml
  + documents
    + <document id>
      + <branch>~<language>
        + document.xml
        + files containing binary part data
```

### NOTE

The import structure starts below the "root", thus there should not be a directory called "root" in the import structure.

The only file that is really required is the info/namespaces.xml file, all the rest is optional, including the documents.

## The info/meta.xml file

This file contains some general properties.

Syntax:

```
<?xml version="1.0"?>
<meta>
  <entry key="daisy-server-version">2.0</entry>
  <entry key="export-time">2006-08-01T15:37:03.946+02:00</entry>
</meta>
```

Meaning of the properties (which are all optional):

Property	Purpose
daisy-server-version	The version number of the Daisy repository server from which the export was created. This information can be used by the import tool to do some conversions upon import.
export-time	Timestamp of when the export was created. This is added by the export tool and purely informational.

export-format

The format of the export. Usually this has the value "default", but in case of the [translation management export](#)<sup>1</sup> this has the value "tm".

## The info/namespaces.xml file

This file lists the namespaces that are used by the documents.

Example structure:

```
<?xml version="1.0"?>
<namespaces>
  <namespace name="FOO" fingerprint="20765a57796f4a774912606..." required="true"/>
  <namespace name="BAR" fingerprint="some finger print" required="false"/>
</namespaces>
```

Each namespace is listed with its name and fingerprint. The required attribute indicates whether the namespace is really required in order for the import to succeed. Namespaces that are really required are those of the documents themselves and those used in link-type fields. Non-required namespaces are those used in links in documents (all links outside the link-type fields). Usually the import tool will register all namespaces, unless the user running the import does not have the Administrator role. In that case, the import can continue if there are some non-required namespaces which are not registered.

## The info/variants.xml file

This file lists the branches and languages that are used by the documents.

Example structure:

```
<?xml version="1.0"?>
<variants>
  <branches>
    <branch name="main" required="true"/>
    <branch name="foo" required="true"/>
  </branches>

  <languages>
    <language name="default" required="true"/>
    <language name="bar" required="true"/>
  </languages>
</variants>
```

The required attribute serves the same purpose as for namespaces. The description of the branches and languages can be set by adding a description attribute.

## The info/schema.xml file

This file defines schema types (field types, part types and document types) to be imported. The root tag of this file is <schema>, and it can contain any number of <fieldType>, <partType> and <documentType> children. Field types and part types should be listed before the document types that make use of them.

So the basic structure is:

```
<?xml version="1.0"?>
<schema>
  <fieldType .../>
  <partType .../>
```

```
<documentType .../>
</schema>
```

## Common configuration

Field, part and document types can all have labels and descriptions in multiple locales. For all three, these are defined using child label and description tags. For example:

```
<documentType name="Image" deprecated="false">
  <label locale="" text="Image"/>
  <label locale="nl" text="Afbeelding"/>
  <description locale="" text="Use this document type to upload images in the Daisy Wiki." /
  >
  <description locale="nl" text="Gebruik dit documenttype om afbeeldingen [...]"/>
</documentType>
```

## Defining a field type

Syntax:

```
<fieldType name="..."
  valueType="string|date|..."
  multiValue="true|false"
  hierarchical="true|false"
  aclAllowed="true|false"
  allowFreeEntry="true|false"
  loadSelectionListAsync="true|false"
  deprecated="true|false"
  size="0">
  [ optional labels and descriptions ]
  [ optional selection list ]
</fieldType>
```

The name and valueType attributes are required, the rest is optional.

For a listing of the possible values for the valueType attribute, see the table later on.

There can optionally be a selection list defined. For this, use one of the following three elements.

### Static selection lists

```
<staticSelectionList>
  <item value="...">
    <label locale="..." text="..." />
    [ nested <item> elements in case of hierarchical list ]
  </item>
</staticSelectionList>
```

The value should be correctly formatted corresponding to the valueType of the field type. The formats are listed in a table further on.

### Query selection lists

```
<querySelectionList query="..."
  filterVariants="true|false"
  sortOrder="ascending|descending|none" />
```

## Link query selection lists

```
<linkQuerySelectionList whereClause="..." filterVariants="true|false"/>
```

## Hierarchical children-linked query selection lists

```
<hierarchicalQuerySelectionList whereClause="..." filterVariants="true|false">  
  <linkFields>  
    <linkField>[field type name]</linkField>  
  </linkFields>  
</hierarchicalQuerySelectionList>
```

## Hierarchical parent-linked query selection lists

```
<parentLinkedSelectionList whereClause="..." parentLinkField="..." filterVariants="true|false"/>
```

## Defining a part type

Syntax:

```
<partType name="..."  
  mimeType="..."  
  daisyHtml="true|false"  
  deprecated="true|false"  
  linkExtractor="...">  
  [ optional labels and descriptions ]  
</partType>
```

Only the name attribute is required.

## Defining a document type

Syntax:

```
<documentType name="..." deprecated="true|false">  
  <partTypeUse partTypeName="..." required="true|false" editable="true|false"/>  
  <fieldTypeUse fieldTypeName="..." required="true|false" editable="true|false"/>  
  [ optional labels and descriptions ]  
</documentType>
```

The partTypeUse and fieldTypeUse elements can be used zero or more times.

## The info/retired.xml file

This file lists documents that should be marked as retired during import.

Syntax:

```
<?xml version="1.0"?>  
<retiredDocuments>  
  <document id="1167-DSY" branch="main" language="default"/>  
</retiredDocuments>
```

The <document> element can be repeated any number of times.

The branch and language attributes are optional and default to main and default respectively.

## The info/collections.xml file

This file lists collections that should be created during import. Collections used in documents are not required to be listed in this file, they will be automatically created if missing. The main purpose of this file is to create additional collections.

Syntax:

```
<?xml version="1.0"?>
<collections>
  <collection>...</collection>
  [ more collection elements ]
</collections>
```

## The documents directory

The documents directory contains all the documents to import. For each document there should be a subdirectory named after the ID of the document. This directory in turn contains again subdirectories for each variant of the document. The name structure is <branch name>~<language name>.

An example structure:

```
documents
+ 123-DSY
  + main-default
    + document.xml
  + main~nl
    + document.xml
+ 124-DSY
  + main-default
    + document.xml
```

The variant directory contains at least a document.xml file, and possibly more files for the part data (if any).

This is the minimal structure for the document.xml file:

```
<?xml version="1.0"?>
<document type="...">
  <name>...</name>
</document>
```

Everything else described below is optional.

## Specifying the owner

The owner is defined by its login in an attribute called owner on the root tag:

```
<document type="..." owner="piet">
  ...
```

## Specifying the version state

For the case where the import would cause a new document version to be created, the state for that version can be specified with a versionState attribute:

```
<document type="..." versionState="draft|publish">
  ...
```

## Specifying the reference language

The reference language can be optionally specified on the root tag:

```
<document type="..." referenceLanguage="en">
  ...
```

## Specifying fields

All fields are defined inside a fields element:

```
<document ...>
  <fields>
    <field .../>
  </fields>
</document>
```

## Single-value fields

```
<field type="..." value="..." />
```

Type attribute contains the name of the field type.

The format of the values depends on the value type of the field, and is described in a table further on.

## Multi-value fields

```
<field type="...">
  <value>...</value>
  <value>...</value>
</field>
```

## Single-value hierarchical fields

```
<field type="...">
  <hierarchyPath>
    <value>...</value>
    <value>...</value>
  </hierarchyPath>
</field>
```

## Multi-value hierarchical fields

Repeat the hierarchyPath element multiple times.

```
<field type="...">
  <hierarchyPath>
    <value>...</value>
    <value>...</value>
  </hierarchyPath>
  <hierarchyPath>
    <value>...</value>
```

```
    <value>...</value>
  </hierarchyPath>
</field>
```

## Specifying parts

All parts are defined inside a parts element:

```
<document ...>
  <parts>
    <part .../>
  </parts>
</document>
```

The syntax for the part element is as follows:

```
<part type="..." mimeType="..." dataRef="..." fileName="..."/>
```

The type attribute contains the name of the part type.

The mimeType attribute specifies the mime type of the data. For example, for Daisy-HTML parts this will be "text/xml". For a PNG-image it is "image/png".

The dataRef attribute specifies the name of a file containing the data of the part. This file should be located in the same directory as this document.xml file. You are free to choose the file name.

The fileName attribute is optional and specifies the fileName property of the part (which is the file name that will be presented to the user when downloading the content of this part).

## Specifying links

Example syntax:

```
<document ...>
  <links>
    <link>
      <title>...</title>
      <target>...</target>
    </link>
    [ more link elements ]
  </links>
</document>
```

## Specifying custom fields

Example syntax:

```
<document ...>
  <customFields>
    <customField name="..." value="...">
    [ more customField elements ]
  </customFields>
</document>
```

## Specifying collections

Example syntax:

```

<document ...>
  <collections>
    <collection>[collection name]</collection>
    [ more collection elements ]
  </collections>
</document>

```

## Field value types and formats

valueType	format
string	as is
date	XML Schema format For example: 2006-07-14T00:00:00.000+02:00 (time component will be ignored)
datetime	XML Schema format For example: 2006-07-18T22:23:00.000+02:00 Note that maximum precision for time component supported by Daisy is seconds.
long	sequence of digits, without separators
double	sequence of digits, use dot as decimal separator
decimal	same as double
boolean	true or false
link	daisy:<docid>@<branch>:<lang> <branch> and <lang> are optional (means: same as containing document, this is recommended as the links will automatically adjust when copying data between branches and languages) <docid> is in the form <sequence number>-<namespace> Examples: daisy:123-DSY daisy:123-DSY@main:default daisy:123-DSY@:default (specifies only language, not branch)