[EN] 02. Importing Metadata from the XML Format

> ⚠ **Note**
>
> The programming library for that extension is `dlibra-app-extension-mf-xml`.

The functionality of the Editor and Administrator Application can be extended with the use of so-called extensions. For example, extensions which make it possible to import metadata to the dLibra system from external sources can be added. The extension for importing a bibliographic description from a file in the XML format is an extension of that type (for information about that format, see here).

The extension for importing metadata from the XML format is based on XQuery queries (for information about XQuery, see here. For the explanations below, we assume that the reader knows the XQuery standard.

The import of bibliographic descriptions will only be possible if the XML extension is properly configured. The default configuration of that extension allows the user to import a bibliographic description from the RDF and MASTER formats.

The XML extension is configured with the use of two property files (property files contain key=value pairs).

- `tests.properties` – a file containing XQuery queries which test the type of the file from which the bibliographic description is to be imported; and
- `conversion.properties` – a file containing the rules of the conversion of an XML file to metadata in the dLibra system.

The editor should create the files on the editor's local drive or make them available via www, and then indicate their location in the configuration of the extension for importing XML files.

The two files are closely interconnected: for every test from file `tests.properties`, there exist conversion rules in file `conversion.properties`. For an XML file with metadata, the importing mechanism performs, one by one, the test queries from file `tests.properties`. If the mechanism encounters a question which returns more than 0 values, it imports the metadata from the XML file with the use of conversion rules (from file `conversion.properties`) which correspond to the test..

In file `tests.properties` , there are XQuery questions which check if the given file with metadata can be imported with the use of the conversion rules associated with that query. The key identifies the conversion rules in file `conversion.properties`.

For example, let us assume that we have the following files (in the example, the default configuration of the extension is used):

`the tests.properties` file:

```
master=for $x in fn:doc({document})/*[fn:compare(fn:name(), 'msDescription')=0] return $x
rdf_dc=for $x in fn:doc({document})/*[fn:compare(fn:local-name(), 'RDF')=0] return $x
```

`the conversion.properties` file:

```
master.Title=for $x in fn:doc({document})//msHeading/title return $x
master.Creator=for $x in fn:doc({document})//msHeading/author return $x
master.Description=for $x in fn:doc({document})//msContents/overview return $x
master.Publisher=for $x in fn:doc({document})//msContents/respStmt/resp/name return $x
master.Contributor=for $x in fn:doc({document})/msDescription/msContents/respStmt//resp return $x
master.Date=for $x in fn:doc({document})//msHeading/origDate return $x
master.Type=for $x in fn:doc({document})//physDesc/form return $x
master.Identifier=for $x in fn:doc({document})//msIdentifier/country/settlement/repository/idno return $x
master.Source=for $x in fn:doc({document})//msPart//idno return $x
master.Language=for $x in fn:doc({document})//msContents/textLang return $x, for $x in fn:doc({document})
//msContents/textLang/@otherLangs return $x
master.Rights=for $x in fn:doc({document})//msIdentifier/repository return $x

rdf_dc=for $x in fn:doc({document})//*[fn:local-name()='Description']/* return $x
```

In the example, file `conversion.properties` contains conversion rules which correspond to tests from file `tests.properties`. There are two types of rules in file `conversion.properties`:

1. the rule for a particular attribute – the key of such a rule consists of a key identifying a test from file `tests.properties`, a period, and an attribute identifier (RDF name) in the dLibra system; in the case of such a rule, the values of all nodes returned by an XQuery query will be assigned to the attribute with the given identifier; and
2. the universal rule – the key of such a rule is the same as a key identifying a test from file `tests.properties`; in such a case, every node returned by an XQuery query must have the same name as the RDF name of the target attribute; XMNLS name spaces are ignored.

In the example, the configuration for the MASTER format is defined with the use of rules of the first type, and for the RDF format – of the second type, but both types can be used in a definition of one format – in such a case, if the universal rule generates values for an attribute which also has a dedicated rule assigned to it, then all values obtained from both rules will be assigned to the attribute.

Only one universal rule can be defined for a given XML format, and only one dedicated rule can be assigned per attribute. However, an XQuery rule may consist of several rules connected by commas, as is the case for the `Language` attribute.

Every XQuery query should make use of a sequence of the {document} sequence of characters to determine the document on which the query is to be performed. The extension changes that sequence of characters to an XML file.

Let us assume that we want to import file A which contains metadata in the XML format. The import mechanism checks, one by one, the tests in file `tests.properties`. The first test which will return a result with a list of values greater than 0 will determine the rules of conversion.

Let us assume that test is the test with the “master” key. The import mechanism chooses the conversion rules form the `conversion.properties` file – all those which begin with the word “master”. Next, values from the XQuery queries are assigned to the appropriate attribute, for example, all values from query `for $x in fn:doc({document})//msHeading/title return $x` will go to the attribute with the Title RDF name.