

# [EN] 01. Importing Metadata from the MARC Format



## Note

The programming library for that extension is [dlibra-app-extension-mf-marc](#).

## General information

By default, attributes from the MARC 21 communication format are imported on the basis of the configuration built into the extension. Instead of the default configuration, a configuration defined in external text files can be used. Their format is a simple set of properties. In order to indicate new configuration files for the extension, its configuration should be displayed (see the [image below](#); displaying the configuration window is described in the [configuration](#) section). After the configuration has been displayed, the source of the new configuration (a file on the drive or a URL address) should be indicated and the "Use the configuration from the given source" option should be selected. Additionally, the character encoding for the MARC files to be imported should be specified. MARC character encoding depends on the information technology system from which such a file is imported.

Import metadanych z formatu MARC - konfiguracja rozszerzenia

## Format pliku marcImport.properties

```
Title=245:${a} ${b} ${n};130;210;222;240;246;730;740;
en.Title=210;222;240;246;730;740;
Creator=100;110;111;
Subject=
Description=6XX;
Publisher=260a;260b;260f;
Contributor=700;710;711;
Date=260c;
Type=
Identifier=920;856u;
Source=
Language=041;546;008/35-37;
Relation=250;534;440;490;800;810;811;830;
Coverage=
Rights=506;540;"PAN";
```

The box above contains the content of a sample `marcImport.properties` configuration file. In that file, the MARC configuration can be defined, that is, the values of elements from the MARC format can be assigned to attributes in the dLibra system.

In every line in the configuration file, there is an import value configuration for one attribute. In every line, on the left side of the equal sign, there is the RDF name of the attribute to which the values from the MARC elements on the right side of the equal sign will be assigned. If the RDF name is preceded by a language name and a period (for example, `en.Title=210;222;240;246;730;740;`), then the rule following the name will be used for importing the metadata for that language. If there is no language name, then the rule is used for importing the metadata for the language selected by the user (in the appropriate tab in the metadata editor). The language name must be a two-letter code compliant with the ISO 639 standard. The RDF names of attributes can be found in the Administrator Application (the attribute edition panel). The elements which can be imported from the MARC format can be, for example, subfield values or particular characters from control fields. If any attribute RDF name is missing from the configuration file, then the field number list next to that attribute will be empty. In such a case, no value will be imported for that attribute.

The basic syntax of MARC field numbers (the list on the right side of the equal sign), from which the attribute values are to be imported, is as follows: `AAAb;`, where `AAA` is a three-digit field number, and `b` is the identifier of a subfield. It is also possible to combine subfields in a notation or to retrieve a range of characters from control fields (those cases are described below). It ought to be noted that the `;` character (semicolon) is a necessary part of a field notation in a configuration.

A subfield value can be left out. Also, multi-value identifiers can be used. Here is some more detailed information about the configuration, with examples:

- `100;` – an example of how a field number can be used.  
In the case of a special field, that notation will cause the value of that field to be imported to the given attribute (it should be remembered that special fields in the MARC format, that is, fields with a value smaller than 010, do not have subfield identifiers). In the case of other fields, that notation will cause all values from all subfields of that field to be imported to the given attribute.
- `260c;` – an example of how a field number with a subfield identifier can be used.  
In the case of such a notation, only the values of a particular subfield of the given field (in this case, `c` of field 260) will be imported.
- `6XX;` – an example of how a multi-value identifier can be used.  
In the case of such a notation, all values of the fields and subfields from the 600–699 range will be imported. It will not be possible to specify particular subfields. Also possible is notation `65X`, which will cause the field values from the 650–659 range to be imported.
- `245:${a} ${b} ${n};` – an example of how MARC field subfields can be combined into one value.  
That notation can be divided into two parts separated by the `:` character (colon):
  1. `245` – this is the number of the field the subfields of which will be combined into one value.
  2. `${a} ${b} ${n}` – this is a template which defines the way in which the subfields will be combined.  
Notation `${a}` means that it is to be replaced with the value of subfield `a` from the field with the number given before the `:` character – in this case, field 245. Therefore, it means that subfields `a`, `b` and `n` will be combined into one value and will be space-separated. For example, if subfield `245a` has value the *first value*, subfield `245b` has value the *second value*, and subfield `245n` has value the *third value*, then that notation will result in value *the first value the second value the third value*. If we want the subfields to be separated by any other character or sequence of characters, we just have to type them in, for example: `245:${a}-${b} subfield n: ${n};`.

\\

(for example,

`245:${a}${b}\\;${n};`

). Polish diacritics and other characters outside of the standard ASCII set must be converted to UTF-8 codes in the “\uXXXX” format, where the “x” characters are hexadecimal digits (the converter is available at: <http://rishida.net/tools/conversion/> – the JavaScript escapes part).

- `008/35-37` – only concerns control fields and means taking that range of characters from a control field.  
That notation consists of two parts separated with the `/` (slash) character:
  1. `008` – the number of the control field from which values will be taken.
  2. `35-37` – the range of characters which will be taken from the field with the number given before the slash.  
Characters 35, 36 and 37 from field 008 will be the value of that notation. If field 008 has character `p` in position 35, character `o` in position 36, and character `l` in position 37, then the value of that notation will be `pol`. If we only want to take one character from a particular position, we just have to specify that position after the `/` character, for example, `008/30`.
- `"PAN";` – that is a constant value added to an attribute regardless of the content of the loaded MARC file. Constant values must be put in straight double quotation marks: `"`. Just like in the case of [templates for combining subfields](#), some characters must be preceded with two slashes:

\\

: this time, the characters are `"` (double quotation mark) and `\` (slash). Also, Polish letters and other non-standard characters must be converted to UTF-8 encoding.

## marcImpRemChars.properties file format

```
end-245b=a|b  
begin-245a=0S/2  
end-260c=c
```

The box above contains a sample configuration file `marcImpRemChars.properties`.

That file makes it possible to define the character sequences to be removed from particular MARC subfields before they are imported to a bibliographic description. Characters (or character sequences) can be removed from the beginning (begin) or end (end) of a MARC subfield. Character strings are defined with the use of regular expressions. The regular expression which can be used in the mechanism for removing characters from MARC values must be compatible with the regular expressions used in the Java language. (for details, see [here](#)).

Line `end-245b=a|b` means that character `a` or character `b` will be removed from the end (the word “end”) of subfield 245b (if, off course, there is any of these characters at the end of the value of subfield 245b). The minus sign separates the determination of the place from which characters are to be removed (in this case, “end”) from the subfield from which the characters will be removed (in this case, 245b). The equal sign is followed by a specification of a regular expression (in this case, `a|b`) which defines what signs are to be removed.

Let us analyze the following example: `begin-245a=ab`. That notation will cause the Editor Application to remove the `ab` sequence of characters from subfield 245a – if such a sequence is found at the beginning of the value of that subfield. Thus, if field 245a in a MARC file has value `abBajki`, then the character removal mechanism will change it to value `Bajki`, and that value will be imported to the bibliographic description.

## Default Configuration

By default, the extension is configured with the following files:

- `marcImport.properties`:

```
Title=245;130;210;222;240;246;730;740;  
Creator=100;110;111;  
Subject=  
Description=6XX;  
Publisher=260a;260b;260f;  
Contributor=700;710;711;  
Date=260c;  
Type=  
Identifier=920;856u;  
Source=  
Language=041;546;  
Relation=250;534;440;490;800;810;811;830;  
Coverage=  
Rights=506;540;
```

- `marcImpRemChars.properties`  
That file is empty by default.