

---

# Set Parameters and Migrate Your Data to Koha 2.2

Paul POULAIN <paul AT koha-fr.org>

Copyright © 2004, 2005 Paul Poulain

This document is related to Koha and is licensed to you under the GNU General Public License version 2 or later (<http://www.gnu.org/licenses/gpl.html>).

Koha-related documents may be reproduced and distributed in whole or in part, in any medium physical or electronic, as long as this copyright notice is retained on all copies.

You may create a derivative work and distribute it provided that you:

1. License the derivative work with this same license, or the Linux Documentation Project License (<http://www.tldp.org/COPYRIGHT.html>). Include a copyright notice and at least a pointer to the license used.
2. Give due credit to previous authors and major contributors.

Commercial redistribution is allowed and encouraged; however, the author would like to be notified of any such distributions.

No liability for the contents of this document can be accepted. Use the concepts, examples and information at your own risk. There may be errors and inaccuracies, that could be damaging to your system. Proceed with caution, and although this is highly unlikely, the author(s) do not take any responsibility.

All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

2005-04-18

Revision History

Revision 2.2.2                      2005-04-18                      rsh

Revision 2.2.0                      English translation by Regula Sebastiao H.  
2005-01-06                      pp

Version initiale

## Table of Contents

1. Introduction .....	2
1.1. General remarks .....	2
1.2. Essentials .....	2
2. First iteration .....	3
2.1. dumpmarc.pl .....	3

2.2. UNIMARC parametrization .....	3
2.3. First import .....	6
2.4. First results .....	6
2.5. Enhancing the cataloging framework .....	8
2.6. Migration of a non-iso2709 file .....	14

# 1. Introduction

## 1.1. General remarks

Data migration is done in an iterative way :

- Examination of data to migrate (supposedly from a ISO2709 file)
- Do part of the parametrization
- Import records and control
- Refine parametrization of bibliographic records
- Re-import record, etc.
- As soon as the import of the bibliographic records is finished and validated, migrate the authority records, which is done with a similar iterative process.

## 1.2. Essentials

The data migration demands librarian's and computer competences . Don't start if you cannot reunite both. If you are IT, you may disappoint your users. If you are a librarian all which follows may seem completely obscure to you.

So if you do not have both competences, it might be a good idea to ask for external help.

Technical essentials:

- Koha installation; it is supposed that Koha is installed. \$KOHA is the directory in which the professional (librarian's) interface is stored.

```
ls -l $KOHA
```

must output: `cgi-bin htdocs modules scripts`

- The configuration file of Koha is in `/etc/koha.conf`
- All following commands are executed from a console (shell) on the server.
- In the shell (command line), verify that Perl can find your modules:

```
export PERL5LIB=$KOHA/modules
```

## 2. First iteration

### 2.1. dumpmarc.pl

Go to the directory where the file of the records to migrate is. In this example it's called `notices.iso`.

To examine its content, the script `dumpmarc.pl` is useful:

```
$KOHHA/scripts/misc/dumpmarc.pl -f notices.iso
```

The whole of the records is displayed in a form which can be read by a normal human being, which is not the case of a record in raw "iso2709" format.

It's possible to stop the display with **CTRL-C** if you have a lot of records!

Examine the records displayed, and note the most frequently used tags. The following tags probably occur most frequently (in UNIMARC, in [...] the corresponding tags of MARC21):

- 200, with title and authors. [245]
- 205, for edition informations. [250]
- 210, publisher. [260]
- 215, physical description [300]
- 010, ISBN [020]
- 011, ISSN [022]
- 600 to 699 subject authorities
- 700 to 799 author authorities

But this document is not meant to be a UNIMARC [nor MARC21 ;-)] lesson!

It's also helpful to look at the item records. They should be stored in Tag 995 if your records respect the recommendation of 995.

### 2.2. UNIMARC parametrization

When your Koha was first installed, you had to choose and to import all UNIMARC tags and sub-fields [valid for other MARC formats too]. You thus can go to

```
Koha >> Parameters >> Biblio framework (MARC structure)
```

This is where you can adapt your frameworks according to your wishes and according to what you have been looking at in the last section.

#### 2.2.1. Basic notion : MARC based / non-MARC based

One of the major constraints of the development team of Koha is to develop a "multi-MARC" application.

But although all declinations of the MARC format have the same form (ISO2709), the meaning of each tag and it's sub-fields can be completely different from one dialect to another.

For example, in UNIMARC, the title is stored in 200\$a.

In MARC21, it's in 245\$a.

Whereas the 245\$a of UNIMARC contains ... nothing (the field simply doesn't exist !)

Well, this means that it's impossible to administer all the differences in a simple way.

The Koha development team has thus chosen a method to circumvent this problem, without putting a strain on performance: everything is stored twice:

### **Note**

To those who wonder about the fact of storing everything twice: the cost in MB's of hard disk is ridiculous compared to the gain in search performance! By the way, and without entering into details, the data actually are stored three times, still for performance reasons. in "MARC" format (200\$a which ignores the meaning of the field) and in a "decoded" form (title which ignores the MARC position).

A table for parametrization is set up at installation. This table contains all the labels for all MARC21 (or UNIMARC) fields and subfields, as well as the links between the two databases.

It's the application which takes care of storing all data twice, namely at the creation or modification of a record.

Even if this adds a lot to the complexity of the operations for record creation/modification, it simplifies the operations of search and reading just as much. Since in an ILS over 90% of access operations are for search and less than 10% are record creation/modification operations, the speed of the application is reinforced even more so.

For example, for the display of a search result of a query, the application uses the "title" directly. If the MARC format was used, the whole record would have to be searched in order to only extract it's title!

## **2.2.2. Biblio framework configuration**

Note that Koha is capable of managing different frameworks. I recommend refining the first framework before creating subsequent ones.

At first, we are not looking at authority records.

Select the default framework and modify it's fields one after the other.

For each MARC tag, the following information has to be defined:

- Repeatable or not. If repeatable, a sign appears in front of the tag, allowing to repeat the field if necessary.
- Mandatory or not. If a field is mandatory, it is not possible to validate the record without at least one subfield in this tag.
- The label of the tag is also defined in the configuration tables. It's possible to adapt the label to your needs, default values are UNIMARC labels..

For each subfield, the following information has to be defined:

- Activated or not. Chose a tab other than -1 (ignore) to activate it. Note that all subfields of a given tag must appear in the same tag. Otherwise, and if the tag is repeatable, Koha won't know how to react when the field is to be repeated!
- Mandatory or not. If the subfield is mandatory, the record can be validated only after containing data.
- Repeatable or not. If a subfield is repeatable, this can be simply done by separating the repeated values by the sign | .
- Linking the "Koha field" with the MARC subfield. As Koha is multi-MARC, the meaning of certain specific MARC fields has to be "taught" to it first. For example, teach it that the subfield 245\$a contains the title of the record, the subfield 245\$c the author(s), etc. In the default template, the main "links" have already been activated. It might be necessary to modify certain links as subject.bibliosubject which point towards a subfield 6XX, or the links to additionalauthors.authors, which contains the link to the co-authors, or the link to bibliosubtitle.subtitle containing the subtitles.

The "Koha fields" can originate from the following tables: biblio, biblioitems, items, additionalauthors, bibliosubtitle, bibliosubject.

- "Related fields". If you search a certain field, Koha will automatically expand the search to the "related fields" which you have declared. This allows to extend an author search to co-authors, to author authorities, if you use them, or to expand a title search to subtitles, serial titles, uniform titles, etc.
- Check or not option "hidden". This option is normally used for all \$9 fields of authorities (see section on authorities). If you check this option, the field will be displayed in the MARC editor, but not for display. This option can be used to hide a subfield from OPAC display.
- Check or not option "URL". If you check this option, the field will be a hyperlink.

There are also 3 options to add constraints for the data input in subfields:

- Authorised values
- Thesaurus
- Plugin

These elements can be neglected at first, they'll be treated later.

Now, go to the cataloging menu and add an empty record. Verify if you have indeed activated all the options you want to.

### 2.2.3. Verification of the "MARC <=> non-MARC link"

The links between the two databases MARC and non-MARC can be verified easily through

```
Koha >> Parameters >> Links Koha - MARC DB
```

This window lists all the connections of the fields between the non-MARC and the MARC fields/subfield.

The links can be modified in this interface, but beware: **each modification is valid for ALL frameworks.**

## 2.3. First import

Return to the shell, and start the first import:

```
$KOHA/scripts/misc/bulkmarcimport.pl -d -c UNIMARC --file notices.iso
```

The option `-d` allows to delete existing records. It's useless for the first import, but as it will be useful later, it's good to start with the right habit.

The option `-c UNIMARC` indicates how the characters of your file are ENCODED. Attention: it's possible to have records in UNIMARC format using characters according to MARC21 norms. If at looking at the records, you see strange diacritical characters, try the other option (`-c MARC21`).

## 2.4. First results

### 2.4.1. Visual verification

As soon as the script has finished running or if you have interrupted it, you can have a look at the catalogue.

Normally, the first result won't look great. Search for a frequently used term and check whether there are any item records (in the simple view as well as in the full view all items are regrouped in a special "group").

If the item records are missing go back to parametrization of Koha so as to make item records appear.

Verify also if both the "simple", i.e. non-MARC, and "complete", i.e. MARC, view contain data.

If subfields seem to be missing in the MARC view on display, two things are possible:

- The subfield exists in the catalogue, but doesn't appear on screen. This means that the subfield has not been "activated", but contains `-1` (ignore) in the tag. Modify the parametrization and go back to check the result at the display without re-importing the data.
- The subfield is actually missing from the database, but you think it should be there. Check with the utility `dumpmarc.pl` if it really exists where you expect it.

If data appears in the "complete" (MARC) record but not in the "simple" (non-MARC) record, this can be OK, or not.

Actually, the number of fields which can appear in the "non-MARC" part is limited. Only the following fields can be found there:

- author
- title
- unititle (uniform title)
- notes (bibliographic)
- abstract
- seriestitle

- copyrightdate
- volume
- number (of volume)
- classification (local classification code)
- itemtype (document type)
- url
- isbn and issn
- dewey
- publicationyear (date of publication, edition)
- publishercode (name of publisher)
- volumedate
- volumeddesc (description of volume)
- illus (illustrator)
- pages
- bnotes (2nd note field)
- size (textarea, to be able to have an area as big as 24x30)
- lccn (Library of Congress classification)

Alls these zones are NOT repeatable in the "non-MARC" part of the database, the part which appears in the display of the simple view. If one of these zones is repeated, only the first one will be displayed or in certain cases, the zones will be separated by the | character.

Three further zones are repeatable:

- Additional authors (additionalauthors.author)
- Subtitles (bibliosubtitle.subtitle)
- Subjects (biblisubject.subject)

## 2.4.2. Verification query

It's also possible to make some SQL queries for data verification.

The following query will display all the fields and subfields with their use. This query is quite helpful in detecting errors of the parametrization of the cataloguing framework:

```
SELECT tab, tagfield, tagsubfield, count( * ) AS tot FROM marc_subfield_structure  
LEFT JOIN marc_subfield_table ON marc_subfield_table.tag=marc_subfield_structure.t  
AND marc_subfield_table.subfieldcode=marc_subfield_structure.tagsubfield  
GROUP BY tab, tag, subfieldcode
```

### 2.4.3. Iterations

It's possible to iterate the imports of the catalogue as long as needed, i.e. until a satisfying result is obtained.

Note that the script `bulkmarcimport.pl` does not modify the bibliographic records.

A migration can by the way be a good occasion to clean up your data, deleting parts which will not be used anymore, or to modify others to render them norm compliant.

To modify the `bulkmarcimport.pl` script needs good knowledge of Perl, and mainly in MARC record manipulation.

The manipulation of MARC records is done by the Perl package `MARC::Record`.

The description can be found with Perldoc:

```
perldoc /usr/lib/perl5/site_perl/5.8.3/MARC/Record.pm
```

(the path is the path of my Mandrake 10.0 machine, it can differ according to your linux/unix distribution)

The same documentation can be found on the net:

<http://marcpm.sourceforge.net/MARC/Record.html>  
[???<http://marcpm.sourceforge.net/MARC/Record.html>]

and

<http://marcpm.sourceforge.net/MARC/Field.html>

`MARC::Record` allows any manipulation directly on the record. It's only flaw is the complex interface which is a direct consequence of the complexity of the iso2709 norm.

## 2.5. Enhancing the cataloging framework

Up till now, all the subfields have exactly the same input format: the subfield is unrestricted, the cataloguer can input any kind of data.

Koha has three more formats for subfield input:

- List of authorized values
- Thesaurus/authority
- Plugin

### 2.5.1. List of authorized values

The lists of authorized values allow to define a list of possible values for a given subfield. This is especially wanted for subfields where only a well defined and limited set of normalized values are allowed for input.

Examples:

- Languages (language codes)
- Countries (country codes)
- Codes for the function of secondary authors

Here follows the example of how to activate the list of language codes.

As soon as this has been done, the subfield isn't a free text zone anymore, but a drop-down list of pre-defined values.

## Note

From an ergonomic point of view, a drop-down list is useful only if the number of given values is limited. In general, the list shouldn't contain more than around 20 entries. We recommend not to use larger lists.

### 2.5.1.1. Definition of the drop-down list box and its authorized values

Koha >> Parameters >> Authorised values >> New category

Choose the category code. It's helpful to use an intelligent code (e.g. LANG for the languages).

After having entered the category code, enter the first possible value.

There are always two elements to the authorized values: the code which is put into the record; and it's label which will be displayed in the cataloguing template.

Example: languages - *eng* would be the code and *english* the label. The code *eng* will be inserted in the record if the cataloguer chooses "english" in the list.

### 2.5.1.2. "Connection"

When your list is complete (or as soon as you wish, it's always possible to edit the list afterwards), return to the parametrization of the biblio framework, go to the field you want to "constraint" (e.g. languages, 041 in MARC21 - but this is still not a MARC class)

In the input constraints for the tag, you have now in the list of "authorized values" a category LANG. Choose this category and from now on the cataloging template doesn't offer a text field in 041 anymore, but a drop-down list of the predefined values.

### 2.5.1.3. Two more things for drop-down lists

... you should know about:

- If a subfield is optional, the MARC editor adds automatically the value empty. If the subfield is mandatory, no empty value is added and has not to be entered. This means that in any case, an empty entry is useless!
- Please note that the values will be listed in the order of the labels, and not in the order of codes. A blank is considered as "smaller than the letter 'A'". Knowing this, it's possible to define a default value for a subfield: enter a space before a label, and the value will be listed on top. Also, as a solitary blank in front of a character string is ignored in HTML, the label will display without the added leading space!

#### 2.5.1.4. Conclusion

Now that you know how to define authorized values, it's time to learn that there exists a script which is actually taking over the task to define the list of languages.

The script is :

```
$KOHA/misc/migration_tools/buildLANG.pl
```

Execute it without parameters so as to have the options (and don't forget to export PERL5LIB so that the script uses certain particular Koha modules).

#### 2.5.1.5. Special case : itemtypes and branches (IMPORTANT)

The list of authorized values for a subfield "linking" is automatically augmented for two particular types.

**These two authorized values must be linked to MARC fields.**

**itemtypes**      The table of document types which are possible in the programme. This table is used in different places.

- The readers can limit their search for a document type
- The loan rules depend on the document types

This table must thus be linked. In UNIMARC, it should logically be connected to the 200\$b field.

The field to which the table itemtypes is linked must also be linked to the *Koha field* biblioitems.itemtype.

**branches**      The table of library branches. At least one branch (the main branch) has to be defined.

This branch must be linked to two subfields of the item record field. For UNIMARC, it's used according to the 995 recommendation.

These two subfields have to be linked to the fields items.holdingbranch and items.homebranch.

### 2.5.2. Thesaurus / Authority lists

I will not explain to you the nature of authority records according to the UNIMARC norm.

Koha is able to manage authorities in the MARC (UNIMARC or another MARC declination) format. The following section tells how to parametrize the different categories of thesaurus/authorities, and how to migrate a thesaurus.

#### 2.5.2.1. Initial remarks

In the UNIMARC norm, subfield \$3 of the bibliographic record is used to stock the number of the authority record, and thus the link between the authority record (A) and a bibliographic record (B) is established.

Koha works with this field, but it uses subfield \$9 to preserve internally the links between records (B and A). \$9 is reserved for the local system, thus it's all conform to the norms. And it presents also some

other internal advantages.

Note that in this section, the migration from a version 2.0 to a 2.2 is not mentioned. It's a complicated migration, please contact me, if you need help as I do have some scripts I could send you.

### 2.5.2.2. Definition of thesaurus / authority list

At the installation of Koha, you had the possibility to import the definition of UNIMARC authorities (at the end, when you had the option to import SQL files).

If this has not been done, do it manually. Search for the SQL file in the directory where you have decompressed Koha.

This parameter file cannot be used directly: it lists all basic fields needed to define your own authority structures.

Follow the example of how to define the structure for authorities for personal author names. The idea is the same for all authorities, no matter whether subjects or authors.

### 2.5.2.3. Parametrization of authority frameworks

Go to: Koha >> Parameters >> Thesaurus Structure

Click on *Add an authority type*.

Input:

- A code for this authority type. For personal names, you could use PN
- Description: of the authority type. This description is purely for information.
- Summary: enter here the elements which will allow to define the display of the records for search result lists. It's possible to define here the subfields which will be displayed in [ ]. Each subfield can be preceded or followed by a string. For example, for personal names of authors, you may indicate (in UNIMARC) :

```
[200a][200b][200c]  
[400a][400z]  
[100a]
```

The display of the summary of the record in search result lists would thus contain the heading field and the see from tracing fields. If you find the list too long add only the line of the [200]s.

- Summary (MARC21):

```
[100a][...][...] to be completed for MARC21  
[400a][...]  
[500a]
```

- The number (000 - 999) of the authority record field corresponding to the bibliographic record field. For the PN authorities, e.g., it's the 200 tag which corresponds to the bibliographic record (tags 700, 701 or 702 UNIMARC, or 700, 710, 720 MARC21). As you can see, you indicate the whole field, not subfields. Actually, Koha will automatically make all subfields automatically correspond in the bibliographic record. In the above example, 200\$a corresponds to 700\$a, 200\$b to 700\$b, etc.

As soon as you have finished, validate the input. The authority type should appear in the list of authorit-

ies. Click on "MARC structure". Koha will detect first that there is no template for this authority type yet and it'll ask which existing template has to be copied. At the beginning only "default" can be copied.

Now, an authority framework can be defined exactly as the framework for cataloging bibliographic records. Some options which are specific to bibliographic records will not be available, but otherwise it looks pretty much the same.

Once the authority framework is defined, go to the authority menu and create an authority record to control how the input template for authorities looks.

#### 2.5.2.4. Connexion B => A

Now that authority records are parametrized, return to parametrization of bibliographic records so as to link the B records (bibliographic) with the A records (authorities).

Thus, return to

```
Koha >> Parameters >> Biblio framework (MARC structure)
```

Go to tag 700 containing the heading of the main author or of the main authors.

**Subfield \$9**      As explained above, Koha uses subfield \$9 to link record B with record A.

Verify if subfield \$9 has been created, and if not, create it, and activate it in the same tab as the other subfields of tag 700. Check the box "Hidden" so as to have it not displayed on the OPAC.

**Link**              Modify the subfield 700. For one of the subfields (logically it would be \$a, but this is not mandatory) select the value PN from the authority/thesaurus list as an input constraint.

Thus you indicated to Koha that the 700 field contains values of the PN authority list.

Add a new record now. Note that before 700\$a three dots are displayed: ...

These ... are a hyperlink opening a popup window, which allows a search in the authority list and to copy over the found values. After selecting the value, the popup window closes and the selected entry has been copied automatically into the bibliographic window.

#### 2.5.2.5. Automatic reconstruction

It is impossible to propose a script which works always. Especially as it would have to be different for PN (personal names), NC (nouns), CO (corporate entries), etc.

Nevertheless you'll find a script to reconstruct NC authorities in the directory `misc/migration_tools/build6xx.pl`, which rebuilds the 606 tag while checking for the presence of subfield \$x.

This script can be used as a base to reconstruct the 700, 500, or whatever you'll need.

#### 2.5.3. Plugins (so far - UNIMARC only)

Plugins are supplementary modules which can be used to treat data in different ways so as to ease cataloging of bibliographic (or authority) records.

For example, for UNIMARC, there are plugins facilitating:

- input of publishers and series statements
- input of coded fields (1XX)

### 2.5.3.1. General remarks

A plugin is active on one given subfield. When the plugin is active, the subfield has the following properties:

- Manual input is possible as for any normal field.
- It has ... which open a window. This popup window assists in filling in the field.
- If you enter or leave a subfield, certain plugins will modify the content of the entered values automatically in other subfields of the cataloging framework.

### 2.5.3.2. Coded fields

In version 2.2 of Koha, the "Ecole Nationale Supérieure des Mines de Paris" has developed a plugin for all 1xx (UNIMARC) which are coded fields.

The ... open up a popup window which contains a drop-down list for each coded element of a subfield, easing thus the cataloguing process considerably.

### 2.5.3.3. Publishers/Series statements

The plugins called `unimarc_field_210c.pl` and `unimarc_field_225a.pl` will fill in automatically the fields containing publishers and the series statement according to the ISBN (if it exists).

This plugin is pretty delicate in its configuration.

First of all, it uses ... authority files. The publisher's list can indeed be looked at as an authority list (= authority being the heading under which the editor is listed).

Thus:

Koha >> Parameters >> Authority values

Add an authority type which must be call EDITORS. It's summary will be:

```
[ 200a ][ / 200b ]
```

and the linking field will be 200.

Define for the publisher authority records the following structure:

```
200$a => ISBN  
200$b => Publisher  
200$c (repeatable) => Series
```

Then "link" the UNIMARC subfields 210\$c and 225\$a.

That's it.

It's now possible to input the "publisher records" by using the ISBN's first two elements (e.g. of ISBN 1-22-333333-4, use 122, but WITHOUT the hyphens).

Enter the information you desire for the publisher, and the series statements (separating them by | as always when separating subfields).

These functions are pretty handy for libraries who have a limited number of publishers (i.e. mainly specialized libraries).

As in cataloguing, after having entered the ISBN in a bibliographic record, going to the publisher field will lead to an automatic entry of the field.

Clicking on the ... in front of the series statements will open a drop-down list with the "series" of this publisher.

Please note that it's not yet possible to create a series statement on the go, and neither to create authority tables automatically from records (in the process of a migration, e.g.), but somebody will do that one day.

## 2.6. Migration of a non-iso2709 file

This case is really easy to manage: construct a MARC::Record (i.e. iso2709) on the go.

This is pretty complex. Especially as all cases must be imagined!

### 2.6.1. Migration of a Texto file

I have some routines to migrate a Texto database. They are not available here as each Texto installation has its own list of fields.

But the use of a Texto file can be summarized as follows:

```
$/ = "";
while (<AJOUT_PILOTE>)
{
    my @fichier = split/\n/, $_;
    # print "taille fichier". $#fichier. "\n";
    foreach my $ligne (@fichier) {
        # on examine le contenu de la fichier
        if ($ligne eq "") {# separateur de fichiers
        }

        my @mots = split(/\./, $ligne);
        $mots[0] =~ s/ //g;
        $last=$mots[0] if $mots[0];
        $resul{$last} .= $mots[1]. " ";
    }

    # ok, on a l'enregistrement, on construit le MARC::Record.
    my $newRecord = MARC::Record->new();
```

This part of code will build a hash table for each record. From there an iso2709 record can be created, and thus imported normally.

```
$resul{ISBN} =~ s/-//g;
my $newField = MARC::Field->new(
    '010', ' ', ' ',
    'a' => $resul{ISBN},
```

```
);  
$newRecord->insert_fields_ordered($newField);  
$newField = MARC::Field->new(  
  '011', '', '',  
  'a' => $resul{ISSN},  
);  
newField = MARC::Field->new(  
  '200', '', '',  
  'a' => $resul{TIT},  
  'b' => $resul{TYP},  
  'e' => $resul{STI},  
  'f' => $resul{AUT},  
  'g' => $resul{NOT},  
);  
$newRecord->insert_fields_ordered($newField);  
my ($bibid,$oldbibnum,$oldbibitemnum) = NEWnewbiblio($dbh,$newRecord,'') unless ($
```

Please note that the above code doesn't treat item records (this is really much too specific a case).