USER-DOCUMENTATION

Conversion between ALEPH-standard-formats of documents

Release: 18, ...
Last Update: 18.01.2008

TABLE OF CONTENTS

1 Classification and limitation

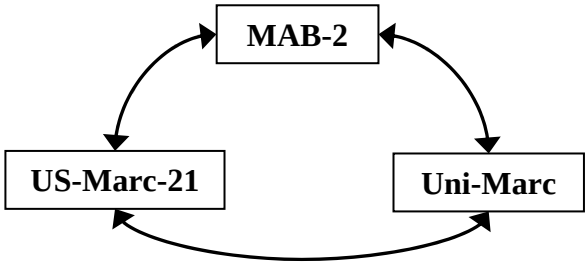
While migrating bibliographic records from third-party systems via Z39.50, the wish arose to convert these data automatically to the format used in the own system.

In principle, fix routines are available for the modification of bibliographic records. Those routines are connected via the table ‚tab_fix' located in the ‚tab-directory' of the corresponding library. By the name of the fix routine a group of entries in ‚tab_fix' will be run through, which edit the bibliographic record that is availabe in the internal ‚doc-format'.
By which entries in the tables, configuration settings or even fixed program-calls the processing of the desired fix routines will be activated depends on the structure and the mode of operation of the corresponding superior system, e.g. the Z39.50-interface.

The description of the superior system and the integration of the fix-routine-program described in this document shall **not** be the main subject of this document. However, under point **2** some examples for main fields of application are listed.

This document describes a fix-routine-program, which converts the existing bibliographic record by an own table. The aim of that program is to convert the format of the data structure rather than modifying some fields a little.

Besides programs for general use, special programs, which serve for the conversion between the three commonly used formats in librarianship, are provided:



In principle, this method can also be used for other fix-routines as long as the desired modification of the doc-record is beneficial realisable through the converting mechanism (which can be quite different from the „re-formatting" described above).

Due to the naming of the programs of that fix-routine (fix_doc_convit_...) we will talk about

"convit"-programs / "convit"-fix-routine

in the following, where "convit" is a synonym for "Conversion between ALEPH-standard-formats".

2 Examples for the integration of the convit-fix-routine

2.1 Converting US-MARC-21 to MAB2 via Z39.50

For the call via Z39.50, the USMARC library in tab_base.eng must be defined as MAB library (ext04). In the Gate configuration, the definition for the remote library remains USMARC.
When receiving a message (Z58-IN-RECORD-CREATE) "vir_z00_z39_usmarc" will further be called, however, the conversion (Z58-IN-RECORD-FIX) has to be made to MAB.
The result is a MAB record in a MAB Z39-Library (ext04).

$data_gate/usmab_z39.conf:

```
...
############################################################
#Present Response Settings
############################################################
Z58-IN-RECORD-TYPE           USMARC
Z58-IN-RECORD-CREATE         vir_z00_z39_usmarc
Z58-IN-RECORD-CHAR-CONV
Z58-IN-RECORD-FIX            USMMB
...
```

$data_tab/tab_fix (ext04):

```
...
! 1                 2                                  3
!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
USMMB fix_doc_convit                  FILE=tab_fix_usm2mab
...
```

Here, "**fix_doc_convit**" is the control program of the convit-fix-routine. The mode of action and the composition of the table stated under FILE= will be described in the following!

3 Call and function of the control program

The control program "**fix_doc_convit**" is connected to the corresponding library in the table ‚tab_fix'.

$data_tab/tab_fix:

```
! 1                  2                                   3
!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
fffff fix_doc_convit                    parameter
```

parameter* :** **FILE= *convit-Tabelle
, DEFAULT= *standard-convit-program*

fffff Identifier of the fix-routine

- Common identifiers are:
  USMMB - US-MARC to MAB
  UNIMB - UNI-MARC to MAB
  MBUSM - MAB to US-MARC
  etc.

FILE= Name of the convit-table in the $data_tab-directory

- Composition see point **4**
- Common names for the corresponding format conversions are:
  tab_fix_usm2mab
  tab_fix_uni2mab
  tab_fix_mab2usm
  etc.
- whereas the „2" stands for "_to_", not for MAB2

-- mandatory parameter--

DEFAULT= Indication of the standard program for processing a field when the column programm/parameter in the table ‚FILE=' is empty
Standard= edit_field (see point **5.6**)

Functioning

- Loading of the convit-table FILE= , if it hasn't been loaded to the table-memory yet.
- Creating a working copy of the source record
  - The DOC-NUMBER of this working copy is consecutive, a "virtual" doc-system-number.
  - The virtual DOC-NUMBER allows connected convit-programs of the program column to recognize if a new record is currently edited .
  - For it is quiet possible that the same record will be edited by fix_doc_convit several times, the original system number wouldn't allow the programs to recognize this !!
- Removing of all fields marked as deleted (DOCX-UPDATE="X") in the working copy
- Creating a target record
  - The new target record's DOC-NUMBER is identical to the source record's system number
  - Thus, if needed, the regular system number can be gathered from the new target record.
- Executing all fields of the working copy in the existing sequential order
  - All table rows of ‚FILE=' will be executed in the existing sequential order for a field, where the field code and the source-field code TAB-SOURCE-FIELD (column 1) are matching. The common masking characters for the field code ("#", "!") can be used in TAB-SOURCE-FIELD.
  - The creation of new fields is done by the connected convit-programs of the column 5 (program and parameter)!
- Removing of all fields marked as deleted (DOCX-UPDATE="X") in the new target record
- Overwriting of the source record with the new target record

4 Structure of the convit-table

The table consists of 5 columns

```
...
! COL   1.   5; ALPHA_NUM, UPPER; #;
!               Source Tag & indicator;
!               Source Tag & indicator;
! COL   2.   1; ALPHA_NUM, LOWER;  ;
!               Source Subfield;
!               Source Subfield;
! COL   3.   5; ALPHA_NUM, UPPER; #;
!               Target Tag & indicator;
!               Target Tag & indicator;
! COL   4.   1; ALPHA_NUM, LOWER;  ;
!               Target Subfield;
!               Target Subfield;
! COL   5. 100; ALPHA_NUM, LOWER; _;
!               Program name with arguments;
!               Program name with arguments;
!
! 1   2   3   4               5
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
!
1####           mab2usm_100
2####           mab2usm_200

300   a 23400 a
304   a 1300  a

417#  a 260   b edit_field        MERGE-I
418#    260     edit_field        ADDNEW,SEL=ag,RENAME=aagb
420     515
...
```

Col-1 - Source-Field-Code

- Field-Code with indicator for choosing of those table rows that are processed for a source field
- Common masking characters ("#", "!") for field-codes may be used.

Col-2 - Source-Subfield-Code

- Indication of a subfield-code of the source field
- The precise meaning of this information depends on the convit-programm according to column 5.
- Normally, this means that this subfield will be chosen (the first one if there is a multiple appearance). The remaining subfields will not be of further interest and will get lost.

Col-3 - Target-Field-Code

- Indication of a field code with indicator for a new target field
- The precise meaning of this information depends on the convit-programm according to column 5.
- Normally, this means the field name of the new field. If the Target-Field-Code contains "#"-characters, these places will usually be replaced with the corresponding characters from the source field!

Col-4 - Target-Subfield-Code

- Indication of a subfield-code for the new target field
- The precise meaning of this information depends on the convit-programm according to column 5.
- Normally, this means a renaming of the subfield-code chosen in column 2.

Col-5 - Program and Parameter

- Indication of a convit-program and possibly related parameters
- There has to be at least a blank between program and parameter.
- Parameters without program are forbidden. The string from the first character that is unequal to a blank up to the next blank or the end of row will be interpreted as a program name!
- If this column is empty, the convit-program that is defined under DEFAULT= will be processed (by default: "edit_field").

- The text after the program name will be interpreted as a list of parameters.
- The parameters of the list are separated by commas.
- A parameter of position, i.e a parameter without an introducing key word with equals sign (e.g. TEXT=) must always stand at a given position in the list!
- A parameter of key word, i.e. a parameter with introducing key word and equals sign may stand at any position in the list.
- For a start, the value of the parameter is always a character string.
- The value of a parameter may be put between inverted commas (Cobol-QUOTE-sign). It **must** be put between inverted commas, if:
  - Blanks at the beginning and/or at the end shall be part of the value
  - Commas or equals signs are part of the value, otherwise those characters will be interpreted as classifying elements of the list.
  - If the inverted commas shall be part of the value, a prefixed backslash is necessary. The backslash itself has to be coded with double backslash.
- By a hexadecimal indication of a character in form of **\x*XX*** (whereas ***XX*** have to be two hexadecimal characters 0-9,A-F) in the value, any character can be passed.
  - You should handle this with care!!
- By a hexadecimal indication of a Unicode-value in form of **\u*XXXX*** (whereas ***XXXX*** have to be four hexadecimal characters 0-9,A-F) in the value, any UTF8-Code of a Unicode-value can be passed.
  - You should handle this with care!!

    - You can not perform a global translate to UTF8 with the created record, because the record contains such signs; the current character-set of of the current record must be UTF8 in the moment of working from fix_doc_convit !

- A special case of a parameter value is the selection of field-text. The description is more comprehensive and will be described separately in point **4.2**.

4.1 Standard processing of a table row

The standard processing of a table row, i.e with an empty program column, is equal to the standard processing of the standard-convit-program that is defined at the control program under DEFAULT= .
Hence, the two following rows are identical in their meaning. For compatibility reasons, any other standard program should also be described like here.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc x TTTtt y
CCCcc x TTTtt y edit_field  ADDNEW
```

- Using the Source-Field-Code ***CCCcc*** this table row was chosen for editing the current field. Common masking characters ("#", "!") may be used.
- From this field a new field with the Target-Field-Code ***TTTtt*** will be generated. Positions with the masking character "**#**" will be replaced by the corresponding position of the current field code and **not** by corresponding positions from ***CCCcc*** !!
- If the Target-Field-Code ***TTTtt*** is empty, the complete field code of the current field will be adopted!

Structure of the new field text

- Source-Subfield-Code ***x*** = empty, Target-Subfield-Code ***y*** = empty
  - The field text will be adopted 1:1.

- Source-Subfield-Code ***x*** = not empty, Target-Subfield-Code ***y*** = empty
  - The subfield text of the first matching subfield will be the new field text.
  - If no Target-Subfield-Code is specified, the subfield text will be adopted **without** subfield label and, thus, the new field is a fixed field without a subfield structure!!
  - If no subfield ***x*** can be found or the subfield text is empty, no new field will be generated.

- Source-Subfield-Code ***x*** = not empty, Target-Subfield-Code ***y*** = not empty
  - The subfield text of the first matching subfield ***x*** will be saved as subfield text of the new subfield ***y*** in the new field text.
  - If no subfield ***x*** can be found or the subfield text is empty, no new subfield and consequently no new field will be generated.

- Source-Subfield-Code ***x*** = empty, Target-Subfield-Code ***y*** = not empty
  - The whole field text will be saved as subfield text of the new subfield y in the new field text.
  - Of course, this only makes sense, if the source field is a fixed field without subfield structures!!
  - If the old field text is empty or has a subfield structure, no new subfield and consequently no new field will be generated.

Examples

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
LDR
051      085   a
100#
200a  b 910   c
331#  a
200a  a 910   a
441##   910#
```

- Adoption of the LDR-field 1:1
- Adoption of the fixed field text of field 051 as subfield a of field 085
- Adoption of the field 100 1:1 with any indicator
- Adoption of the first subfield b from field 200a to subfield c of field 910
- Adoption of the subfield text of the first subfield a from field 331 with any indicator as fixed field text from the homonymous new field
- Adoption of the (first) subfield a from field 200a to subfield a of field 910
- Adoption of all 441-fields with any indicator as field 910 with indicator identical to the first position of indicator of the source field; the field text will be adopted 1:1 .

4.2 Parameter value for selecting a field text

The parametrisation of a program often requires the specification of a certain text area of a doc-sequential record. E.g. to copy it or to check it.
Usually, one or more of the following indications may be necessary:

- Subfield code
- OCCUR-indication for the subfield
- relative position within a subfield text or fixed field text
- maximum length as from the relative position
- field code
- OCCUR- indication for the field (e.g. in case of repeatable fields)
- Selecting the field from input-/output-record, the output-/target-record or using the field text from the current "work space" of the processed field

If you need all these values for specifying the text because due to actual program requirements, you don't want to use fixed default values, then you can do it with a maximum of 7 individual parameters (with fix_doc_convit you cannot use parameter lists).

We therefore have created a syntax which can include all indicators mentioned above. Wether the program in question will actually use all of them, is defined by the program. The program will also ignore redundant indicators or write an error to the logfile if neccessary as well as choose default values if indicators are missing.
If a parameter allows the selection of field text in this way, it will be mentioned in the parameter description of the program,
e.g. "Selection of field text according to ..." , "***fieldtext-selection***" , "***fieldtext-sel***"
This form is not yet integrated to the parameters of escpecially older programs because the requirements arose only later.

In the following section you will find the description of the general syntax, including some examples.

General Rules:

- The individual elements can be separated with one or more blanks, e.g. to improve readability.
- „Default“ means the way a program behaves if the respective parameter is not given.

A fieldtext-selection can be a selection of text from the „actual“ field independent from a certain field code, a selection of text from a selected field or just the selection of a field..

fieldtext-selection	**::=**	***text-selection***
	or	***text-selection / field-selection***
	or	***/ field-selection***

text-selection

Selection of text from a field text

- In general this means the indication of a subfield code.

Default= How the missing of a text selection is interpreted by the program depends on the program itself: Maybe the complete field text is used or just the first subfield.

field-selection

Selection of a field from a doc-record

- This takes place before the selection of text.
- But because a text selection is typically always specified and field selection is rather the exception this combination has been choosed.

Default= The data of the „actual“ field will be used for text selection.

text-selection	**::=**	***sf***
		or ***sf*** **[** ***pos*** **:** ***len*** **]**
		or ***sf*** **[** ***pos*** **:** **]**
		or ***sf*** **[** ***pos*** **:** ***len*** **]**
		or ***sf*** **[** ***pos*** **:** **]**
		or ***sf*** **{** ***sf-occur*** **}**
		or ***sf*** **[** ***pos*** **:** ***len*** **]** **{** ***sf-occur*** **}**
		or ***sf*** **[** ***pos*** **:** **]** **{** ***sf-occur*** **}**
		or **ALL**
		or **FIELDTEXT**

sf

Subfield-code (1 character)

- The character "**#**" can be used as a wildcard for any subfield.
- The character "**$**" is not allowed !
- The field data must have a subfield structure.

pos
len

Using ***pos*** and ***len*** you can specify a substring from field text without a subfield structure or from subfield text. A substring selection will be indicated with square brackets.

- ***pos*** gives the relative position from the beginning of the string starting with 0 for the first character.
- Is the specification of length ***len*** missing, the substring will be built from *pos* to the end of the field text or the subfield text.
- A specification of ***sf*****[0:]** is formally correct but makes no sense.
- The specification of ***pos*** and ***len*** may consist of up 4 digits. The following conditions have to be met:
  1 <= ***len***
  pos + ***len*** <= 2000

sf-occur

OCCUR-specification for subfield. This means the *sf-occur*'th occurence of the subfield in the complete selected fieldtext will be used.
Default= The first found field with *fieldcode* will be used.

ALL
FIELDTEXT

The complete field text will be selected, regardless whether the field has a subfield structure or is a fixed field.

field-selection	**::=**	***fieldcode***
	or	***fieldcode*** **{** ***field-occur*** **}**
	or	***fieldcode*** **:** ***field-record***
	or	***fieldcode*** **{** ***field-occur*** **}** **:** ***field-record***

fieldcode Fieldcode (1-5 characters)

- Ordinary masking characters such as "#" and "!" can be used.

field-occur OCCUR-specification for field..
Default= The first field matching *fieldcode* will be used.

field-record Specification of record for field selection

`I` | `INP` | `INPUT` - Field will be searched in input record.

`O` | `OUT` | `OUTPUT` - Field will be searched in output record generated so far..

Standard= INPUT

sf-occur	**::=**	***occur***
field-occur	**::=**	***occur***

occur Position subfields/fields to select in context of all appropriate subfields/fields. An OCCUR-selection will be indicated with curly brackets.

- Specification of keyword
- Specification of keyword in combination with 1-4 digits
- Specification of an absolute position beginning from the front
- Specification of an absolute position beginning from behind
- A specification of a position ***pppp*** consists of 1-4 digits with 1 <= pppp <= 2000

Werte für ***occur***

F	-	the first object.
F*pppp* *pppp*	-	the pppp'th object from the beginning.
FNL FIRST-NOT-LAST	-	the first but not the last object
FNL*pppp* FIRST-NOT-LAST*pppp*	-	the pppp'th object from the beginning but not the last object.
L	-	the last object.
L*pppp* -*pppp*	-	the pppp'th object from behind.
LNF LAST-NOT-FIRST	-	the last but not the first object
LNF*pppp* LAST-NOT-FIRST*pppp*	-	the pppp'th object from behind but not the first object.

Examples:
The examples are formally correct but may make no sense in the context of MAB or MARC.

`[6:1]/008`

- Field selection: field 008 from the input record
- Text selection: 1 character starting at position 6 (means: the 7th character) from fixed field text

`a[6:1]{L}/328##{L2}:OUTP`

- Field selection: the last but one occurance of field 328 irrespective of indicator from the so far produced fields of the new output record
- Text selection: 1 character starting as position 5 form the last subfield a.

`#{L}/528##{2}:I`

- Field selectionl: the 2nd occurance of field 528 irrespective of indicator form inpurt record.
- Text selection: Subfield text of last subfield.

5 Description of generally usable convit-programs

Overview

5.1	chkit_const	-	Checking by comparing a selected text with constant text
5.2	chkit_contains	-	Checking the appearance of constant text in selected text
5.3	const_field	-	Creating a field with constant field text
5.4	cut_transl	-	Cutting and translating of selected text
5.5	cyclic_fieldno	-	Generating of cyclic field numbers
5.6	edit_field	-	General field editing • Standard program if no convit-program specified (if no other program is defined as standard by DEFAULT=) • Standard entry for DEFAULT= in the control program
5.7	str_cat	-	Appending text to an existing field
5.7			
5.8	get_sub_all	-	Merge of all/selected subfields to one subfield

5.1 chkit_const - Checking by comparing a selected text with constant text

This program compares selected fieldtext with constant text. Is the comparison successful/true the processing continues with the standard program edit_field. Is the comparions not successful/negative/false the processing of the table row will be canceld without effect.

The checking program has no effect on the state of the data. In addition to the parameters described here all parameters for edit_field can be used.
The text to be compared does not need to have any association with the data selected in column 1 and 2. It may descend from the input record or from the fields generated already. The comparison just implements a conditional processing with edit_field.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc x TTTtt y chkit_const  parameter
```

parameter **:** same as for program **edit_field** (see **5.6**)
, CHK-SRC=*fieldtext-selection*
, CHK-OP=*comp-op*
, CHK-TXT=*const-str*
, CASEIGN=*sw*
, WHICH=*keyword*
, NOFLD-OK=*sw*
, NOSF-OK=*sw*
, NOTXT-OK=*sw*

- The meaning of column 1 – 4 is equal to the standard described in point **4.1**.
- If the comparison is successful the processing continues with program edit_field.

CHK-SRC= Specification of source text to be compared using fieldtext selection according to point **4.2**
If the parameter WHICH= is used, a possibly present OCCUR specification has no meaning for the field selection and is therefore not allowed in this context.
-- mandatory --

CHK-OP= Relational operator for text comparison of source text (1.operator) and constant text (2.operator)

- `EQ` – equal
- `NE` – not equal
- `GT|HT` – greater
- `GE|HE` – greater or equal
- `LT` – less
- `LE` – less or equal

-- mandatory –

CHK-TXT= Text constant as 2.operator for comparison
-- mandatory --

CASEIGN= Y/N-Switch
Ignore case for comparison
- applies only to 26 basic letters
- does not apply for special characters (e.g. German „umlauts“)

Default= N

WHICH= Specification of keyword for determination of fields given in CHK-SRC= that will be testet for a complete result
- A specification of ***field-occur*** in CHK-SRC= for field text selection is not allowed!!

ALL	-	all applicable fields
ONE-OFF-ALL ONE	-	At least one field must match
FIRST F	-	the first applicable field (maybe also the only one)
FIRST-NOT-LAST FNL	-	the first applicable field but only if it is not the last applicable field also (this means: there are 2 applicable fields at least)
LAST L	-	The last applicable field (maybe also the only one)
LAST-NOT-FIRST LNF	-	The last applicable field but only if it is not the first applicable field (this means: there are 2 applicable fileds at least)

Default= utilization of ***field-occur***-specification from field text selection; if not there the first applicable field will be used.

NOFLD-OK= If there is no applicable field according to the parameter CHK-SRC=, the result of the comparison is always false and processing of the field will be canceld. With NOFLD-OK=Y set this yields to a true result and processing will be continued.
Default= N

NOSF-OK= **WHICH=ALL**:
If there ist no subfield to be found according to ***text selection***, the testing of this field ist false by default and hence the complete result is false also. With NOSF-OK=Y the testing of this certain field will be assessed as true and the complete result can still become true
otherwise:
If no subfield can be found in all field texts to be considered according to ***text-selection***, the result of the testing is false and processing will be canceled. With NOSF-OK=Y the result will be assessed as true and processing will continue.
Default= N

NOTXT-OK= **WHICH=ALL**:

If there is no substring according to **[*pos*:*len*]** of ***text-selection*** to be found in the field text of a certain field, the testing of the fiel dis false and hence the complete result is false also. This is only possible if ***pos*** is greater than the length of the subfield text or of the constant field text. With NOTXT-OK=Y the testing of this single field will be assessed as true and the complete result can still become true.

otherwise:

If no substring according to **[*pos*:*len*]** of ***text-selection*** can be found in all field texts to be considred, the result of the testing is false and processing of the field will be canceled. This is only possible if ***pos*** is greater than the length of the subfield text or of the constant field text. With NOTXT-OK=Y the result will be assessed as true and processing will continue.

Default= N

5.2 chkit_contains - Checking the appearance of constant text in selected text

This program checks the existence of a search string in the selected field text of one or more fields. If this leads to a true result the processing will be continued with program edit_field, otherwise the processing of the table row will be canceled without effect.

- The field text selection will be determined and tested by **one** field if:
  - No field selection with **CHK-SRC=** , the field text from the actual buffer will be used.
  - Field selection with **CHK-SRC=**, but without **WHICH=**ALL or **WHICH=**ONE-OF-ALL
- The field text selection will be determined and test by **several** fields if:
  - Field selection with **CHK-SRC=** without ***field-occur*** and **WHICH=**ALL or **WHICH=**ONE-OF-ALL

The distinction between uppercase and lowercase can be suppressed. The complete ("primary") result of the test can be negated. The nonexistence of a field, subfield or substring can be regarded as a true result (Parameter NO...-OK=), (bear in mind the **differing** impact of NOSF-OK=/NOTXT-OK= with WHICH=ALL!)
Additionally the characters "*****" and "**?**" can be interpreted as wildcard-characters in the search string.

The complete result (not considering the effects of NEGATE=/NO...-OK=) is **true** if the following applies:

- Checking selected text from **one** field:
  - Selected text according to *text-selection* in CHK-SRC= is existing and the search text is contained in selected text
  - The complete result can be modfied by NEGATE= and all NO...-OK= .
- Checking selected text from **several** fields with WHICH=ONE-OF-ALL:
  - Selected text according to *text-selection* in CHK-SRC= is existing in **at least one** field and search text is contained in selected text.
  - The complete result can be modified by NEGATE= and all NO...-OK=.
- Checking selected text from **several** fields with WHICH=ALL:
  - For **all** fields the result ist rue:
    - Selected text according to *text-selection* in CHK-SRC= is existing and the search text is contained in selected text
    - Does a subfield according to *text-selection* in CHK-SRC= not exist, NOSF-OK=Y has to be specified.
    - Does a substring according to *text-selection* in CHK-SRC= not exist, TXT-OK=Y has to specified.
  - The complete result can be modified with NEGATE= and NOFLD-OK=.

In all other cases the complete result is **false**.

The checking program has no effect on the state of the data. In addition to the parameters described here all parameters for edit_field can be used.

The text to be compared does not need to have any association with the data selected in column 1 and 2. It may descend from the input record or from the fields generated already. The comparison just implements a conditional processing with edit_field

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc x TTTtt y chkit_contains  parameter
```

***parameter* :** same as for program **edit_field** (see **5.6**)
, CHK-SRC=*fieldtext-selection*
, CHK-TXT=*const-str*
, WHICH=*keyword*
, WILD= *sw*
, CASEIGN=*sw*
, NEGATE= *sw*
, NOFLD-OK=*sw*
, NOSF-OK=*sw*
, NOTXT-OK=*sw*

- The meaning of column 1 – 4 is equal to the standard described in point **4.1**.
- If the comparison is successful the processing continues with program edit_field .

CHK-SRC= Specification of source text to be compared using fieldtext selection according to point **4.2**
If the parameter WHICH= is used, a possibly present OCCUR specification has no meaning fort he field selection and is therefore not allowed in this context.
-- mandatory --

CHK-TXT= Search string (constant text); with WILD=Y "*" and "?" will be interpreted as wildcard charcters
-- mandatory --

WHICH= Specification of keyword for determination of fields given in CHK-SRC= that will be testet for a complete result

- A specification of ***field-occur*** in CHK-SRC= for field text selection is not allowed!!

`ALL`	-	all applicable fields
`ONE-OFF-ALL` `ONE`	-	At least one field must match
`FIRST` `F`	-	the first applicable field (maybe also the only one)
`FIRST-NOT-LAST` `FNL`	-	the first applicable field but only if it is not the last applicable field also (this means: there are 2 applicable fields at least)
`LAST` `L`	-	The last applicable field (maybe also the only one)

LAST-NOT-FIRST
LNF - The last applicable field but only if it is not the first applicable field (this means: there are 2 applicable fileds at least)

Default= utilization of ***field-occur***-specification from field text selection; if not there the first applicable field will be used

WILD= Y/N-Switch

The characters star "*****" and question mark "**?**" in the search string will be interpreted as wildcard characters.

"*" - Any string , even with length = 0

"?" - Placeholder for exactly one character (1 byte)

Default= N

CASEIGN= Y/N-Switch

Ignore case for comparison

- applies only to 26 basic letters
- does not apply for special characters (e.g. German „umlauts“)

Default= N

NEGATE= Y/N-Switch

Negates the complete result

- true -> false
- false -> true
- NEGATE= does not have any effect an the meaning of the „NO...-OK=“- parameter

Default= N

NOFLD-OK= If there is no applicable field according to the parameter CHK-SRC=, the result of the comparison is always false and processing of the field will be canceld. With NOFLD-OK=Y set this yields to a true result and processing will be continued.

Default= N

NOSF-OK= **WHICH=ALL**:

If there ist no subfield to be found according to ***text selection***, the testing of a certain field is false by default and hence the complete result is false also. With NOSF-OK=Y the testing of this certain field will be assessed as true and the complete result can still become true

otherwise:

If no subfield can be found in all field texts to be considered according to ***text-selection***, the result of the testing is false and processing of the field will be canceled. With NOSF-OK=Y the result will be assessed as true and processing will continue.

Default= N

NOTXT-OK= **WHICH=ALL**:
If there is no substring according to **[*pos*:*len*]** of ***text-selection*** to be found in the field text of a certain field, the testing of the field is false and hence the complete result is false also. This is only possible if ***pos*** is greater than the length of the subfield text or of the constant field text. With NOTXT-OK=Y the testing of this single field will be assessed as true and the complete result can still become true.
otherwise:
If no substring according to **[*pos*:*len*]** of ***text-selection*** can be found in all field texts to be considred, the result of the testing is false and processing of the field will be canceled. This is only possible if ***pos*** is greater than the length of the subfield text or of the constant field text. With NOTXT-OK=Y the result will be assessed as true and processing will continue.
Default= N

5.3 const_field - Creating a field with constant field text

This program creates a new target field with a constant field text.

The problem is, that this new target field shall normally be created only once per record, independent of a certain Source-Field-Code *CCCcc* . This problem is solved by the parameter ONCE-IDN= („singleness-identifier“).

The value of the parameter is a 1 or 2-digit numerical value greater zero. If a target field for this ONCE-IDN-Value of the current record has already been created, the program will be terminated without effect. By specifying the Source-Field-Code as „**#####**“ the creation of the target field is

- independent of a source field and will always be processed,
- the target field will be created right after the first cycle through the table,
- the target field will be created first of all, if this is the first entry in the table.

If one specifies the same ONCE-IDN-Value during several program calls, one can make the creation of the target field with the constant text dependent of several source fields.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc   TTTtt y const_field  parameter
```

parameter **:** ***field-text***
, ONCE-IDN=*once-num*

- The meaning of column 1, 3 and 4 is equal to the standard described in point **4.1**.
- The indication of the subfield in column 2 is irrelevant for the text selection, the specified field-text will be taken as „chosen“ text

field-text Indication of the constant target field text

- If column 4 is used by target subfield-code, this text mustn’t contain any subfield label.
- Otherwise, the text may contain one or several subfield labels, whereas the proper formal structure won’t be verified!!

Standard= no generation of a new target field

ONCE-IDN= „singleness-identifier“ as a 1 or 2-digit numeric character string
Standard= 00, no „singleness-check“

5.4 cut_transl - Cutting and translating of selected text

This program generates a new field text in the current working area. This field text is formed by "cutting" a text with field text selection, „translating" the this text and storing it as new text in the empty fieldtext buffer of the current working area.

The new field text, if any present, will be further processed with edit_field. Therefore all parameters for edit_field can be used in addition to the paramerters described here. The effect is that the fieldtext from the field given in column 1 will be replaced in the working area and fürther processed with edit_field. Combined with the multiple options of edit_field many tasks can be performed.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc x TTTtt y cut_transl  parameter
```

***parameter* :** same as for program **edit_field** (see **5.6**)
, FROM=*fieldtext-selection*
, TO=*fieldtext-selection*
, CASEIGN=*sw*
, TRANSL=*transl-str*

- The meaning of column 1 – 4 is equal to the standard described in point **4.1**.
- Is the new field text not empty the processing continues with program edit_field .

FROM= Specification of the source text to be cut out and translated with field text selection according to point **4.2**
-- mandatory--

TO= Specification of target for new text with field text selection according to point **4.2**
- Only specification of ***text-selection*** in the form of ***sf*** , [***pos***:***len***] and their combinations are taken into account.
- This means the new text will be stored as a subfield or a fixed field text;.
- if necessary starting at position ***pos*** in full length oder up to maximum length ***len.***

Default= fixed field text from position 0

CASEIGN= Y/N-Switch
Ignore case for comparison
- applies only to 26 basic letters
- does not apply for special characters (e.g. German „umlauts")

Default= N

TRANSL= Translation of text
For decription see chapter below.
Default= no translation, the text remains unaltered

Operation of the TRANSL= -Parameter

Example:
```
TRANSL=";/;ab c/1111;x y z/XYZ;/;"
TRANSL=";std_no_source/;a/1;"
TRANSL=";/;x y/XY;/std_not_found;"
```

- The 1st character (here a semicolon) defines the delimiter for the specific TRANSL commands and has to be used for this purpose only in the value of the parameter. .
- If the semicolon is part of a source or target string definition of a translation, another suitable character must be used.
- The 2nd character (here a slash) defines the delimiter between source and target string definition for one specific TRANSL command. It has to be used for this purpose only in the value of the parameter. If the slash is part of a source or target string definition of a translation, another suitable character must be used.
- In the first command the source string is always empty. This first translation will be used in the following cases:
  - The text selected with FROM= is empty or could not be found.
  - For this situation a default value for the target string can be defined with this 1st translation.
  - If the target string definition is empty too, no field text will be built.
- If the last but not the first TRANSL-command has an empty definition for the source string this is used in the following case:
  - The text selected with FROM= is not empty, but no appropriate source string could be found in the TRANSL-command.
  - In this case the target string defined in this last translate command can be used as default for “other” values.
  - If the target string is empty too, no field text will be built.
- If the target string definition in the last but not first command is not empty and no appropiate source string definition could be found, the text selected with FROM= remains unchanged.

5.5 cyclic_fieldno - Generation of cyclic field numbers

This program generates target field codes starting with the target field code in column 3. If there is no target field code in column 3, the field code of the selected field is the same as of the source field code.

If a new target field code could be generated, the processing will continue with edit_field. Therefore all parameters for edit_field can be used in addition to the paramerters described here. The effect is the same as if the new target field code had been placed in column 3 and only edit_field was called.

The new field code will be generated by repeated („cyclic") addition of a certain amount to the source field code. This cyclic add-up is allowed only a limited number of times. If a field code has been generated for which there is no field in the new target record so far, it becomes the new target field code. For checking the existence of the firld the indicator can be incorporated.
This cyclic addition works normally works on the 3 digits of the numeric source field code but can be restricted to just a part of this field code. The remaining part has to be not necessarily numeric.
If the numerical value of the field code exceeds the value that can be represented by the maximum number of digits in the field code the generation of field codes stopps, even if the possible number of repetitions is not exploited yet.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc x TTTtt y cyclic_fieldno  parameter
```

parameter **:** same as for program **edit_field** (see **5.6**)
, CYC-ADD=*nnn*
, CYC-CNT=*nnn*
, CYC-IND=*sw*
, NUM-TYP= NNN | NNX | XNN | NXX | XNX | XXN

- The meaning of column 1 – 4 is equal to the standard described in point **4.1**. Deviating from this column 2 will be replaced with the generated target field code internally.
- If a new target field code can be generated the processing will continue with edit_field. Kann ein neuer Target-Fieldcode bestimmt werden, wird die Verarbeitung vom Programm edit_field fortgesetzt.

CYC-ADD= Numerical value for cyclic addition
- 1 <= ***nnnn*** <= 500

-- mandatory --

CYC-CNT= Numerical value of max number of iteration of cyclic addition.
- 1 <= ***nnnn*** <= 500

-- mandatory --

CYL-IND= Y/N-Switch
Incorporation of indicator for checking the existence of a field with the generated target field code.
Default= N

NUM-TYP= Specification (of part) of the 3 digit source field code om which the cyclic addition takes place by adding the CYC-ADD= value.

- Numerical positions are marked with "**N**".
- Non-numerical positions are marked with "**X**".

Default= **NNN**

5.6 edit_field - General field editing

This program realises a certain basic functionality for the adoption of field- and subfield contents from the source document to the target document. Hence, this program is used as the default value for the parameter DEFAULT= of the control program, i.e. for the standard program with empty program column.
This means, that this program needn't be entered to the program column, if no special parameters are necessary.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc x TTTtt y edit_field  parameter
```

parameter* :** ***action
, SEL-I=*sel-ind-str*
, RENAME-I=*rename-ind-str*
, SEL=*sel-sf-str*
, RENAME=*rename-sf-str*
, CAT-INP=*catinp-str*
, PREFIX=*prefix-str*
, SUFFIX=*suffix-str*
, CAT=*cat-str*
, CAT-OUTP=*catoutp-str*
, SORT=*sort-sf-str*
, FIRST=*sw*

- The meaning of columns 1-4 is equal to the standard described under point **4.1**.
- The indication of the subfield in column 2 and 4 are irrelevant once the parameter SEL= or RENAME= is not empty!!

action Type of saving the new field in the new doc-record
The existence of a homonymous field in the new doc-record might be considered. For determining this field either the 3-digit field code of the new field without indicator or the field code with indicator are considered. The latter, if the action ends with "**-I**".
By default, the last located field will be used; if FIRST=Y it will be the first located field

ADDNEW , ADDNEW-I
- Save new field at the end of the new record

ADDFIRST , ADDFIRST-I
- Save new field at the end of the new record, if it is the first one with that name
- Otherwise the field will be discarded!!

ADDNEXT , ADDNEXT-I
- Save new field at the end of the new record, if there are homonymous fields
- Otherwise the field will be discarded!!

EXCHANGE , EXCHANGE-I

- Replacing of the last (or first) homonymous field, if there is one
- Otherwise the field will be treated as in ADDNEW / ADDNEW-I

CHGNOADD , CHGNOADD-I

- Replacing of the last (or first) homonymous field, if there is one
- Otherwise the field will be discarded!!

MERGE , MERGE-I

- The field must contain subfields!!
- Appending of the subfields of the new field to the last (or first) homonymous field, if there is one
- Only now, the parameter CAT= will take effect!!
- Otherwise the field will be treated as in ADDNEW / ADDNEW-I

MRGNOADD , MRGNOADD-I

- The field must have subfields!!
- Appending of the subfields of the new field to the last (or first) homonymous field, if there is one
- Only now, the parameter CAT= will take effect!!
- Otherwise the field will be discarded!!

Standard= ADDNEW

SEL-I= Additional selection/verification of indicators

In addition to the selection of the indicator by *CCCcc* (Column 1) more indicators can be verified here. If the current indicator is not correct the program will abort the processing!!

You may create a ' negative selection' by filling in a minus character at the first position. You should assign *CCCcc* with ***CCC##*** then!!

- The first character after the minus character will be used as separator for the indicator details
- Single-digit indicator details are always filled up with blanks. You may use the special characters "**#**" and "**!**
- `Example: SEL-I=-;01;!2;3#;`

Default= no additional selection/verification of indicators

RENAME-I= Renaming indicators

After the indicator has been renamed according to Source-/Target-Field-Code (column-1/ column-3) another renaming is possible by using this parameter. That's why the source indicator should be preserved by assigning ***TTTtt*** with ***TTT##***

You can code several four-digit renamings. The first two characters are used for selection/verification of the indicator and the following two charcters are the new indicator.

- The first character will be used as separator for the different renamings
- A renaming will always be filled with blanks (up to 4 digits)
- For selecting an indicator you can use the special characters "**#**" and "**!**"
- You can use the "**#**" character for the new value of the indicator, which means, this position will be replaced by the old value
- Example: `RENAME-I=:01ab:1#A :2!B#:##ZZ:`

Default= no additional renaming of the indicator

SEL= Selection of subfield

Indication of the subfields as character string, which are to be chosen from the source field; if the first character is a minus "-", all will be chosen except the specified ones

- The field must have a subfield-structure.
- An indication of source/target-subfield sf-*x*/sf-*y* in column 2/4 is irrelevant and will be ignored.

Standard= No selection of subfield

RENAME= Renaming of subfields

Indication of pairs of subfields as character string; the first character is the old Subfield-Code, the second character is the new Subfield-Code

- The field must have a subfield-structure.
- The first character in the pair may also be a "#", i.e. all remaining characters will then be renamed; i.e. such a pair should always be the last two characters!!
- An indication of source/target-subfield sf-*x*/sf-*y* in column 2/4 is irrelevant and will be ignored.

Standard= No renaming

CAT-INP= Merging ("catenate") of subfields, which where composed from data of the Source-Field (from the „INPUT“)
Multiple CAT-commands can be coded consecutively in this parameter, which are processed in this order. The description is more comprehensive and will be described separately.

- The field must have a subfield-structure.
- Unlike the CAT=-parameter, this "catenating" is processed before the execution of the PREFIX=-/SUFFIX=-parameters and without inclusion of old subfields at *action* MERGE/MERGE-I.
- This parameter only makes sense, if subfields should be merged before PREFIX=/SUFFIX= works!

Standard= No merging of subfields

PREFIX= SUFFIX= After the new field text from the source field was build according to Source/Target-Subfield-Code (column-2/4) and the parameters SEL=, RENAME= and CAT-INP= , the Prefix and Suffix will be inserted to all subfields or the fixed field text.

- A Prefix/Suffix can’t be assigned specifically to a particular subfield. Then, only this field may be generated or another solution must be found.

CAT= Merging ("catenate") of subfields
Multiple CAT-commands can be coded consecutively in this parameter, which are processed in this order. The description is more comprehensive and will be described separately.

- The field must have a subfield-structure.

Standard= No merging of subfields

CAT-OUTP= Merging ("catenate") of subfields directly before the output of the new field (from "OUTPUT") but before sorting of subfields by SORT=
It is possible to code several CAT-instructions one by one for this parameter, which are processed in the given order. The description is more comprehensive and will be described separately.

- The field must have a subfield structure
- Unlike CAT=-Parameter with *action* MERGE/MERGE-I all subfields (even the old ones) will be integrated equally in the cat-process
- This parameter only makes sense, if you work with *action* MERGE/MERGE-I

Standardt= No merging of subfields

SORT= Sorting the subfields directly before the output of the new field

- The subfields will be sorted according to the order given by the parameter's value.
- The order of subfields of the same name remains intact.
- If "**#**" is used all remaining, not explicitly stated subfields will be arranged at this position.
- If the parameter value does not contain a "**#**" the position of all subfields which are not included in the order stays intact. Only subfields that shall be sorted will be rearranged among each other, which means, that there may be other subfields between two sorted subfields!!
- Example: `SORT=abcde / SORT=abc#de / SORT=abcde#`

Standard= no sorting of subfields

FIRST= Y/N-Switch

If an already existent field shall/will, according to be included in the processing by *action* , then, by default, the last existent field will be used for determining. Choosing FIRST=Y will take the first located field to work with.

Standard= N

Mode of action of the CAT-INP=/CAT= -parameters

Example: **`CAT=";ab / ;cc, ;uv;#x :: ;## : ;"`**

- The 1st character (here a semicolon) defines the delimiter for the specific CAT-commands and has to be used for this purpose only in the value of the parameter. If the semicolon is part of the delimiter string of a CAT-command, another suitable character must be used.
- Each CAT-command consists of a subfield-code-pair. The following characters, up to the semicolon, will be used as delimiter string for the merge of the subfields.
- 1st CAT-command: append all subfields-a to the **last** existing subfield-b. If there is no subfield-b, the first subfield-a will be renamed to subfield-b and all remaining subfields-a and appended to it. The delimiter is blank-slash-blank.
- 2nd CAT-command: merge all subfields-c to only one subfield-c, i.e. all additional subfields-c will be appended to the **first** subfield-c. The delimiter is comma-blank.
- 3rd CAT-command: analog to the 1st CAT-command but without delimiter; the subfield texts will be directly joined.
- 4th CAT-command: append all remaining subfields to the last existing subfield-x. If there is no subfield-x, the first remaining subfield will be renamed to subfield – x and all remaining subfields will be appended to it. The delimiter is blank-colon-colon-blank. All subfields with a Subfield-Code that hasn't been used explicit in earlier CAT-commands, are the remaining subfields.
- 5th CAT-command: subfields with the same Subfield-Code will be merged to one subfield, analog to the 2nd CAT-command.

Attention !!

- At *action* MERGE/MERGE-I the CAT-parameter will be processed after the merge (i.e. the appendage of the subfields to the subfields of an existing field).
- In doing so, the already existing subfields will always be included in the target search but never used as source-subfield.
- That is, the subfields of an existing field persist and can only be expanded by appending new subfields according to the CAT-commands!!
- In contrast, the CAT-INP=-parameter effects only the newly created subfields but before inserting Prefix and Suffix according to PREFIX=/SUFFIX=.
- The parameter CAT-OUTP= always effects all subfields. The old subfields at *action* MERGE/MERGE-I are neglectable.

5.7 str_cat - Appending text to existing field

This program tries to append the selected text of the source field to the last existing and matching target field according to Target-Field-Code ***TTTtt*** .

If no target field according to ***TTTtt*** (masking allowed) can be found, a new field will be generated in a way as if the program-/parameter-column would be empty and the standard processing takes place.

Otherwise, the result strongly depends on the Source-/Target-Subfield-indication and the structure of the fields (fixed/subfields). Typically, this program should be used for fields with subfield-structure and with an indication of a source/target-subfield-code only!!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc x TTTtt y str_cat  parameter
```

parameter **:** ***delimiter***

- The meaning of columns 1-4 is equal to the standard described under point **4.1**, if no existing new field according to ***TTTtt*** can be located.
- Additionally, the Target-Field-Code can be used for target field search. Hence, masking should be handled with care. The field code of a located matching field may be different to the code, that is generated if no target field was located

delimiter Delimiter-String when appending the selected text to existing text
- This can be a fixed field text as well as a subfield text
- The Source-/Target-Subfield-Codes define which texts are meant.

Standard= no delimiter text, direct joining

Structure of the new field text with located target field

Subject to Source- and Target-Subfield ***x***/***y*** and of the field text structure of the Source- and Target-Field, results following new field text in the last matching target field:

- As appending text the entire field text (Source-Subfield-Code ***x*** = is empty; also field text with subfield-structure) or the subfield text of the first subfield ***x*** will be used. If the text is empty (field text empty, no subfield ***x*** located), the processing will be cancelled.
- The source text will be appended to the entire field text of the target field (Target-Subfield-Code ***y*** = empty or target field with fixed field structure) or to the first located subfield ***y*** with ***delimiter*** . If no subfield ***y*** can be located, a new subfield ***y*** will be created at the end of the target field text.

- You can easily see, that in fields with subfield structure Source- and Target-Subfield-Code ***x***/***y*** should always be coded; otherwise this doesn't make sense and may lead to uncontrollable results.

5.8 get_sub_all - Joining of all/selected subfields to a single subfield

This program generates a new text from the subfield texts of all/selected subfields by joining these texts. If a delimiter is defined, it will be placed between the source texts. Otherwise the texts will be written together directly.
The order of the source field persists and will not be changed according to the selective indication ***subf-sel.***
With this text a new field ***TTTtt*** will be created, either as fixed field text or as subfield ***y*** (Target-Subfield-Code ***y*** = empty/not empty).

A source field without subfield structure will not be processed.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc   TTTtt y get_sub_all  parameter

parameter :   subf-sel
              , delimiter
```

- The meaning of columns 1and 3 is equal to the standard described under point **4.1**.
- The column 2 (Source-Subfield-Code) is irrelevant. A possible value will be ignored.

subf-sel Indication of subfields that should be selected
- If the first character is a minus, all except the following subfield-codes will be selecteda.
- e.g. `"acd" , "-uvb"`

Standard= all subfields

delimiter Delimiter-String between the selected subfield texts
Standard= no delimiter text, direct joining

6 Description of USMARC-to-MAB convit-programs

Overview

6.1 usm2mab_ldr - Processing usmarc-field LDR (Leader)

This program converts the USMARC-field LDR. From the information of this field the fixed MAB-fields "**050**", "**051**" and/or "**052**" are generated or corrected, in case they already exists as new fields.
These fields are treated by other usm2mab-programs, too.

A detailed description of the specific, mostly relating to position, updates and dependencies will not be given here. If required, the precise conversion can be taken from the program!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
LDR                usm2mab_ldr
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant.

6.2 usm2mab_006 - Processing usmarc-field 006

This program converts the USMARC-field 006. From the information of this field the fixed MAB-fields "**050**", "**051**" and/or "**052**" are generated or corrected, in case they already exists as new fields.
These fields are treated by other usm2mab-programs, too.

A detailed description of the specific, mostly relating to position, updates and dependencies will not be given here. If required, the precise conversion can be taken from the program!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
006             usm2mab_006
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant.

6.3 usm2mab_007 - Processing usmarc-field 007

This program converts the USMARC-field 007. From the information of this field the fixed MAB-fields "**050**" and/or "**057**" are generated or corrected, in case they already exists as new fields.
These fields are treated by other usm2mab-programs, too.

A detailed description of the specific, mostly relating to position, updates and dependencies will not be given here. If required, the precise conversion can be taken from the program!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
007                usm2mab_007
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant.

6.4 usm2mab_008 - Processing usmarc-field 008

This program converts the USMARC-field 008. From the information of this field the fixed MAB-fields "**050**" and/or "**051**" are generated or corrected, in case they already exists as new fields.
These fields are treated by other usm2mab-programs, too.

Additionally, the following new fields will be generated, if relevant source data are available:

- "**425a** " Subfield $$**a (**Publication date**)**
- "**037b** " Subfield $$**a** (Language code)

A detailed description of the specific, mostly relating to position, updates and dependencies will not be given here. If required, the precise conversion can be taken from the program!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
008              usm2mab_008
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant.

6.5 usm2mab_440 - Processing usmarc-field 440

This program processes the title's data from the USMARC-fields "440" and "490".

If a subfield $$v is included in the source field, the additional target field "**455** " will be generated out of it.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
440## a 451   a usm2mab_440
490## a 454   a usm2mab_440
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- The Target-Field-Code will be adopted 1:1 and mustn't include "**#**". If necessary, the indicator will be modified by the program.
- Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant!

6.6 usm2mab_700 - Processing usmarc-field 700 (Authors)

This program processes the USMARC-fields containing author's data, usually the fields "700" and "720".
The next free target field for MAB-author-fields "100", "104" up to "196" will be determined. The search starts at the Target-Field-Code of the table entry.
If all target fields already exist, the process will be cancelled.

The program generates a new text for the Target-Subfield by merging the following Source-Subfield-Texts with delimiters:
$$**a** " **<<[**" $$e "**]>>**"

- Only the first Subfield $$e with the action designator will be used

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
100## a 100   a
700## a 100b  a usm2mab_700
720## a 100b  a usm2mab_700
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- The Target-Field-Code determines the search's starting point for the next, not yet existing, new field "100", "104" to "196". The indicator also remains in the specific Target-Field-Codes.
- The Source-Subfield-Code is not analysed and irrelevant!
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. $$a is used by default.

For working with fields in the original language the following parameters can be used:

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc x TTTtt y usm2mab_700  parameter
```

parameter **:** **, LNG-SRC=*fld-code***
, LNG-TRG=*fld-code*

LNG-SRC= Field-Code of field with data in original language

- You can use the special characters "**#**" and "**!**".
- Example.: `LNG-SRC=100##`

Default= No additional editing of a field with original language

LNG-TRG= Field code of target field for storing data in original language

- You can use the special charactes "**#**" and "**!**".
- Esample: `LNG-TRG=A00b`

Default= A00

- The assignment of source field and field in original language ist he result of subfield 6; eg. Sf-6 in the current author field 100##. Then the field with field code LANG-SRC= will be searched which has an identic subfield 6.
- The new text for the target field will be built in the same manner as the source field.
- The target field codes will be generated accordingly to "100", "104", ... , i.e. the 2nd and 3rd character (00, 04, ...) are taken over, the other characters come from LNG-TRG= .

6.7 usm2mab_710 - Processing usmarc-field 710 (Corporate body)

This program processes the USMARC-fields containing the corporate's data, usually the fields "710"/"711", but also "110"/"111".
The next free target field for MAB-corporate-fields "200", "204" up to "296" will be determined. The search starts at the Target-Field-Code of the table entry. If all target fields already exist, the process will be cancelled.

The program generates a new text for the Target-Subfield by merging the following Source-Subfield-Texts with delimiters:
$$**a**" **/** "$$**b** " **<** "$$**d**"**,** "$$c" **>**"

- The text after the subfields $$a/$$b has to be parenthesised even if there are no subfields a/b existent

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
110## a 200   a usm2mab_710
111## a 200   a usm2mab_710
710## a 200b  a usm2mab_710
711## a 200b  a usm2mab_710
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program!
- The Target-Field-Code determines the search's starting point for the next, not yet existing, new field "200", "204" to "296". The indicator also remains in the specific Target-Field-Codes.
- The Source-Subfield-Code is not analysed and irrelevant!
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. $$a is used by default.

For working with fields in the original language the following parameters can be used:

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CCCcc x TTTtt y usm2mab_710  parameter
```

parameter* :** **, LNG-SRC=*fld-code
, LNG-TRG=*fld-code*

LNG-SRC= Field-Code of field with data in original language
- You can use the special characters "**#**" and "**!**".
- Example.: `LNG-SRC=711##`

Default= No additional editing of a field with original language

LNG-TRG= Field code of target field for storing data in original language
- You can use the special charactes "**#**" and "**!**".
- Example: `LNG-TRG=B00b`

Default= B00

- The assignment of source field and field in original language ist he result of subfield 6; eg. Sf-6 in the current corporate body field 200##. Then the field with field code LANG-SRC= will be searched which has an identic subfield 6.
- The new text for the target field will be built in the same manner as the source field.
- The target field codes will be generated accordingly to "200", "204", ... , i.e. the 2nd and 3rd character (00, 04, ...) are taken over, the other characters come from LNG-TRG= .

7 Description of the UNIMARC-to-MAB convit-programs

Overview

7.1 uni2mab_100 - Processing unimarc-fields 100, ...

The program processes the 100's UNIMARC-fields. These information will be used to generate new fields or to correct existing new fields.
The following Source-Fields are processed:

- "**100##**" ➔ "**004** " Subfield $$**a** (creation date)
- "**100##**" ➔ "**037** " Subfield $$**a** (Language)
- "**100##**" ➔ "**051** " fixed field
  - (field "**105##**" exists in Source-record)
- "**100##**" ➔ "**052** " fixed field
  - (field "**105##**" does not exists in Source- record)
- "**105##**" ➔ "**050** " fixed field
- "**105##**" ➔ "**051** " fixed field
- "**105##**" ➔ "**434** " Subfield $$**a**
- "**105##**" ➔ "**437** " Subfield $$**a**
- "**106##**" ➔ "**050** " fixed field
- "**110##**" ➔ "**052** " fixed field
- "**115##**" ➔ "**050** " fixed field
- "**116##**" ➔ "**050** " fixed field
- "**120##**" ➔ "**050** " fixed field
- "**126##**" ➔ "**050** " fixed field
- "**130##**" ➔ "**050** " fixed field
- "**130##**" ➔ "**057** " fixed field
- "**135##**" ➔ "**050** " fixed field

A detailed description of the specific, mostly relating to position, updates and dependencies will not be given here. If required, the precise conversion can be taken from the program!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
1####           uni2mab_100
```

- The Source-Field-Code has the standard meaning. The program checks, if the Source-Field-Code is a relevant field acoording to the above description. Non-matching fields will not be processed.
- "#####" could be entered to the first column, but "1####" will narrow the program calls reasonably. For optical/documentary reasons one could also create a specific line for each relevant field. You shouldn't forget a single relevant field then!
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant!

7.2 uni2mab_210 - Processing unimarc-field 210 (Publishers)

This program processes the repeatable UNIMARC-field 210 containing publishers' data.

If there are no new MAB-fields for the 1. Publisher (neither 410 nor 412), these fields will be created.
If there are new MAB-fields for the 1. Publisher but no fields for the 2. Publisher (neither 415 nor 417), these fields will be created.
If there are fields for the 1. Publisher and for the 2. Publisher, a new repeatable field „418" will be created based on the Publishers' data from the current UNIMARC-field 210.
In addition, a new MAB-field „425" will be creates based on the first located entry regarding the publication date.

Information about 1. Publisher

- If in "210##" Subfield $$a and/or $$c, then:
  - "**210##**" Subfield $$**a** ➔ "**410** " Subfield $$**a**
  - "**210##**" Subfield $$**c** ➔ "**412** " Subfield $$**a**
- If in "210##" Subfield no $$a and no $$c, but $$e and/or $$g then:
  - "**210##**" Subfield $$**e** ➔ "**410a** " Subfield $$**a**
  - "**210##**" Subfield $$**g** ➔ "**412a** " Subfield $$**a**

Information about 2. Publisher

- If in "210##" Subfield $$a und/oder $$c, then:
  - "**210##**" Subfield $$**a** ➔ "**415** " Subfield $$**a**
  - "**210##**" Subfield $$**c** ➔ "**417** " Subfield $$**a**
- If in "210##" Subfield no $$a and no $$c, but $$e and/or $$g then:
  - "**210##**" Subfield $$**e** ➔ "**415a** " Subfield $$**a**
  - "**210##**" Subfield $$**g** ➔ "**417a** " Subfield $$**a**

Information about additional Publishers

- If in "210##" Subfield $$a and/or $$c, then:
  - "**210##**" Subfield $$**a** / $$**c** ➔ "**418** " Subfield $$**a** / $$**g**
- If in "210##" Subfield no $$a and no $$c, but $$e and/or $$g then:
  - "**210##**" Subfield $$**a** / $$**c** ➔ "**418a** " Subfield $$**a** / $$**g**

Information about Publication date

- A target field "425" will be created once based on the first located data.
  - "**210##**" Subfield $$**d** ➔ "**425** " Subfield $$**a**
  - "**210##**" Subfield $$**h** ➔ "**425** " Subfield $$**a**

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant!

7.3 uni2mab_225 - Processing unimarc-field 225 (Common title)

This program processes the repeatable UNIMARC-field 225 containing the common title's data.

The program determines independently which MAB- common title-field "451", "461" to "491" are not present. The search starts with the defined Target-Field-Code of the table.
If all fields already exist, the processing will be cancelled! This will also happen, if the Source-Field doesn't contain a Subfield $$**a!**
From a standard number in Subfield $$x , the fields "452", "462" to "492" will be created.

Information about common title

- "**225##**" Subfields ... ➔ "**451*tt***" Target-Subfield / Subfield $$**a**
- "**225##**" Subfields ... ➔ "**461*tt***" Target-Subfield / Subfield $$**a**
- .....
- "**225##**" Subfields ... ➔ "**491*tt***" Target-Subfield / Subfield $$**a**
- ***tt*** will be taken from the Target-Field-Code of the table
- The Target-Subfield-Text will be generated by merging the following Source-Subfield-Texts with delimiters
  - **$$a " = "$$d " / "$$f " , "$$h " : "$$i " ; "$$v**

Information about standard number

- "**225##**" Subfield $$**x** ➔ "**452a** " Subfield $$**x**
- "**225##**" Subfield $$**x** ➔ "**462a** " Subfield $$**x**
- .....
- "**225##**" Subfield $$**x** ➔ "**492a** " Subfield $$**x**

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
225## a 451b  a uni2mab_225
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- The Source-Subfield-Code is not analysed and irrelevant!
- The Target-Field-Code determines the search's starting point for the next, not yet existing, new field "451", "461" to "491". The indicator also remains in the specific Target-Field-Code. For the generated Target-Fields "452", "462" to "492" always the indicator "**a** " will be set!
- The Target-Subfield-Code determines the Subfield-Code for the new text of the fields "451" to "491". By default, $$a will be used. Subfield $$a will always be generated for the fields "452" to "492"!

7.4 uni2mab_500 - Processing unimarc-field 500

The program processes the UNIMARC-field "500" for generating the uniform title in the MAB-field "304".

The program generates a new text for the Target-Subfield by merging the following Source-Subfield-Texts with delimiters:
$$**a**" **<** "$$**k**", "$$**l**", "$$**m**", "$$**r**", "$$**s**"; "$$**v**" **/** "$$**w**" **>**"

- The text after the subfield $$a has to be parenthesised even if there is no subfield $$a.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
50000 a 304   a uni2mab_500
50010 a 304a  a uni2mab_500
50001 a 304   a uni2mab_500
50011 a 304   a uni2mab_500
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- The Target-Field-Code has the standard meaning, but it isn't analysed in the program!
- The Source-Subfield-Code is not analysed and irrelevant!
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. By default, $$a will be used.

7.5 uni2mab_501 - Processing unimarc-field 501

The program processes the UNIMARC-field "501" for generating the statement of the collective title in MAB-field "300".

The program generates a new text for the Target-Subfield by merging the following Source-Subfield-Texts with delimiters:

$$a" < "$$b", "$$c", "$$k", "$$m", "$$r", "$$s", "$$u
", "$$w", "$$x", "$$y" >"

- The text after the subfield $$a has to be parenthesised even if there is no subfield $$a.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
501## a 300   a uni2mab_501
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- The Target-Field-Code has the standard meaning, but it isn't analysed in the program!
- The Source-Subfield-Code is not analysed and irrelevant!
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. By default, $$a will be used.

7.6 uni2mab_503 - Processing unimarc-field 503

The program processes the UNIMARC-field "503" for generating the uniform title in MAB-field "304".

The program generates a new text for the Target-Subfield by merging the following Source-Subfield-Texts with delimiters:

```
$$a" < "$$b", "$$d", "$$e", "$$f", "$$h", "$$i", "$$j
          ", "$$k", "$$l", "$$m", "$$n" >"
```

- The text after the subfield $$a has to be parenthesised even if there is no subfield $$a.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
5030  a 304   a uni2mab_503
5031  a 304a  a uni2mab_503
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- The Target-Field-Code has the standard meaning, but it isn't analysed in the program!
- The Source-Subfield-Code is not analysed and irrelevant!
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. By default, $$a will be used.

7.7 uni2mab_700 - Processing unimarc-field 700 (Authors)

The program processes the UNIMARC-fields containing the authors data, ususally the fields "700" and "720".
The next free Target-Field for MAB-author-fields "100", "104" up to "196" will be determined. The search starts at the Target-Field-Code of the table entry.
If all target fields already exist, the process will be cancelled

The program generates a new text for the Target-Subfield by merging the following Source-Subfield-Texts with delimiters:
$$**a** " **,** " $$**b** " " $$**c** " " $$**d**

If the current Source-Field-Code = "**702##**" and there is a Source-Subfield $$**4** for coded action determinators, the decoded text will be appended in addition:
$$**a** " **,** " $$**b** " " $$**c** " " $$**d** " **<<[**" $$**4** (decoded) "**]>>**"

The decoding of $$**4** is:

- **070 ➔ Author**
- **080 ➔ Author of Introduction**
- **330 ➔ Dubios author**
- **340 ➔ Editor**
- **440 ➔ Illustrator**
- **560 ➔ Originator**
- **660 ➔ Recipient**
- **730 ➔ Translation**
- others ➔ **Other**

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
700## a 100   a uni2mab_700
701## a 104a  a uni2mab_700
702## a 100b  a uni2mab_700
720## a 100   a uni2mab_700
721## a 104a  a uni2mab_700
722## a 100b  a uni2mab_700
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- The Target-Field-Code determines the search's starting point for the next, not yet existing, new field "100", "104" to "196". The indicator also remains in the specific Target-Field-Codes
- The Source-Subfield-Code is not analysed and irrelevant!
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. By default, $$a will be used.

7.8 uni2mab_710 - Processing unimarc-field 710 (Corporate body)

The program processes the UNIMARC-fields containing the corporate bodies' data, ususally the field "710".
The next free Target-Field for MAB-corporate body-fields "200", "204" up to "296" will be determined. The search starts at the Target-Field-Code of the table entry.
If all target fields already exist, the process will be cancelled

The program generates a new text for the Target-Subfield by merging the following Source-Subfield-Texts with delimiters:
$$**a** " **/** " $$**b** " " $$**c** " **<** " $$**d** "**,** " $$**e** "**,** " $$**f** " **>**"

- The text after the subfield $$a/$$b/$$c has to be parenthesised even if there are no subfields a/b/c.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
710## a 200   a uni2mab_710
711## a 204a  a uni2mab_710
712## a 200b  a uni2mab_710
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program.
- The Target-Field-Code determines the search's starting point for the next, not yet existing, new field "200", "204" to "296". The indicator also remains in the specific Target-Field-Codes.
- The Source-Subfield-Code is not analysed and irrelevant!
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. By default, $$a will be used.

8 Description MAB-to-USMARC convit-programs

Overview

8.1 mab2usm_002 - Processing mab-field 002 (Date)

The program processes the MAB-field 002. These information will be used to generate new fields or to correct existing new fields.
The following Source-Fields are processed:

- "**002##**" ➔ "**008**   " fixed field
  - The open date is extracted from subfield $$a of the field and saved in the relevant position of Target-Field "008".

A detailed description of the specific, mostly relating to position, updates and dependencies will not be given here. If required, the precise conversion can be taken from the program!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
002##          mab2usm_002
```

- The Source-Field-Code has the standard meaning. The program checks, if the Source-Field-Code is a relevant field acoording to the above description. Non-matching fields will not be processed.
- "#####" could be entered to the first column, but "002##" will reasonably narrow the program calls. For optical/documentary reasons one could also create a specific line for each relevant field. You shouldn’t forget a single relevant field then!
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant!

8.2 mab2usm_037 - Processing mab-field 037 (Language codes)

The program processes the 037-MAB-fields. These information will be used to generate new fields or to correct existing new fields.
The following Source-Fields are processed:

- "**037a** " ➔ "**008** " fixed field (insert Language code)
- "**037b** " ➔ "**008** " fixed field (insert Language code)
- "**037##**" ➔ "**0410** " Subfields $$a with specific language codes
- "**037##**" ➔ "**04107**" Subfields $$a with specific language codes

A detailed description of the specific, mostly relating to position, updates and dependencies will not be given here. If required, the precise conversion can be taken from the program!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
037##            mab2usm_037
```

- The Source-Field-Code has the standard meaning. The program checks, if the Source-Field-Code is a relevant field acoording to the above description. Non-matching fields will not be processed.
- "#####" could be entered to the first column, but "037##" will narrow the program calls reasonably. For optical/documentary reasons one could also create a specific line for each relevant field. You shouldn't forget a single relevant field then!
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant!

8.3 mab2usm_050 - Processing mab-fields 050, 051, ...

The program processes basically the 050-MAB-fields. This information will be used to generate new fields or to correct existing new fields.
The following Source-Fields are processed:

- "**050##**" ➔ "**007** " fixed field
- "**051##**" ➔ "**LDR** " fixed field
- "**051##**" ➔ "**008** " fixed field
- "**052##**" ➔ "**LDR** " fixed field
- "**052##**" ➔ "**008** " fixed field
- "**052##**" ➔ "**533** " Subfield $$**n(reproduction note)**
- "**057##**" ➔ "**007** " fixed field

Additionally the following Source-Fields are processed. They are not processed in specific programs, because they have a strong relation to 050-fields and target-fields.

- "**LDR##**" ➔ "**LDR** " fixed field
- "**300##**" ➔ "**LDR** " fixed field

A detailed description of the specific, mostly relating to position, updates and dependencies will not be given here. If required, the precise conversion can be taken from the program!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
LDR##           mab2usm_050
300##           mab2usm_050
050##           mab2usm_050
051##           mab2usm_050
052##           mab2usm_050
057##           mab2usm_050
```

- The Source-Field-Code has the standard meaning. The program checks, if the Source-Field-Code is a relevant field acoording to the above description. Non-matching fields will not be processed.
- "#####" could be entered to the first column, but "050##" will narrow the program calls reasonably. For optical/documentary reasons one could also create a specific line for each relevant field. You shouldn’t forget a single relevant field then!
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant!

8.4 mab2usm_100 - Processing mab-fields 100, 104, ... (Authors)

This program formats the form of heading of authors from the MAB-fields 100##, 104## to 196##.

The source subfield $$a will be splitted to subfield $$a and subfield $$e. The action designator, taken from the last squared brackests, is placed in $$e. The squared brackests and possible non-sorting-characters, in front of or inside the brackets, will not be adopted.
The action designator with brackets and non-sorting-characters will be removed from subfield $$a.

In Source-Field-Code "**100** " the field "**1001** " will be created, in all other Source-Field-Codes and indicators it is the repeatable field "**7001** ".

```
  "100  "  →  "1001 "
  "100b "  →  "7001 "
  "100c "  →  "7001 "
  "104# "  →  "7001 "
...
```

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
1####           mab2usm_100
```

- Only column 1 is relevant for the selection.
- The program itself checks the field code of the passed field for 100##, 104## to 196##. All other fields are not processed. "#####" could be entered to the first column, but "1####" will narrow the program calls reasonably.

8.5 mab2usm_200 - Processing mab-fields 200, 204, ... (Corporate body)

This program formats the form of heading of corporate bodies from the MAB-fields 200##, 204## to 296##.

The source-subfield $$a will be adopted and formatted. If there is text inside angle brackets (not douple-angles, which are non-sorting-characters in ALEPH), these brackets will be replaced.
If Subfield $$a contains the delimiter " **/** " (blank-slash-blank), the field will be splitted at first appearance; the text in front of the delimiter goes to $$a, the text after the delimiter goes to $$b.

In Source-Field-Code "**200** " the field "**1102** " will be created, in all other Source-Field-Codes and indicators it is the repeatable field "**7102** ".

```
  "200  "  →  "1102 "
  "200b "  →  "7102 "
  "200c "  →  "7102 "
  "204# "  →  "7102 "
...
```

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
2####           mab2usm_200
```

- Only column 1 is relevant for the selection.
- The program itself checks the field code of the passed field for 200##, 204## to 296##. All other fields are not processed. "#####" could be entered to the first column, but "2####" will narrow the program calls reasonably.

8.6 mab2usm_425 - Processing mab-field 425 (Year of publication)

The program processes the MAB-field 425. These information will be used to generate new fields or to correct existing new fields.
The following Source-Fields are processed:

- "**425a** " ➔ "**008** " fixed field
  - The first 4-digit numeric string unequal "0000" in Subfield $$a will be interpreted as year of publication and saved at a specific position in target field „008“, in case this position isn't occupied by a year-value.

A detailed description of the specific, mostly relating to position, updates and dependencies will not be given here. If required, the precise conversion can be taken from the program!

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
425##          mab2usm_425
```

- The Source-Field-Code has the standard meaning. The program checks, if the Source-Field-Code is a relevant field acoording to the above description. Non-matching fields will not be processed.
- "#####" could be entered to the first column, but "425##" will narrow the program calls reasonably. For optical/documentary reasons one could also create a specific line for each relevant field. You shouldn't forget a single relevant field then!
- Target-Field-Code, Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant!

8.7 mab2usm_451 - Processing mab-fields 451, 461, ... (Common title in original form)

This program processes the MAB-fields 451##, 461## bis 491##, containing the common title in original form.

The source-subfield $$a will be adopted and formatted. If Subfield $$a contains the delimiter " **;** " (blank-semicolon-blank), the field will be splitted at last appearance; the text in front of the delimiter goes to $$a, the text after the delimiter goes to $$v.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
451#    4900    mab2usm_451
461#    4900    mab2usm_451
...
491#    4900    mab2usm_451
```

- Source-Field-Code and Target-Field-Code have the standard meaning; the current field code will not be analysed by the program.
- That's why the program has to be linked to the relevant fields specifically.
- Source-Subfield-Code and Target-Subfield-Code are not analysed and are irrelevant!

8.8 mab2usm_540 - Processing mab-fields 540, 541, ... (Standard book numbers)

This program processes the fields 540 to 543, containing the standard book numbers.

The source subfield $$a will be adopted and formatted. If the first string without blank (character string until first blank) contains no numeric character,this text will be deleted. Normally, this is the „lead text“ ISBN, ISMN, ISSN or ISRN.

If the next string without blank contains at least one numeric character, this string will be interpreted as the standard book number. This number will be saved to subfield $$a, the remaining text in Target-Subfield $$c.

If the first string contains no numeric character, the complete content from Source-Subfield $$a will be saved to Target-Subfield $$c.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
540     020     mab2usm_540
540a    020     mab2usm_540
540b  a 020   z
540z  a 020   z
541     0242    mab2usm_540
...
543     0242    mab2usm_540
543a    0242    mab2usm_540
543b  a 0242  z
543z  a 0242  z
```

- Source-Field-Code and Target-Field-Code have the standard meaning; the current field code will - with one exception - not be analysed by the program:
  - If the Target-Field-Code = "**022##**", the subfield $$c will not be generated because this is not provided in the USMARC-field 022. The data will get lost.
- That’s why the program has to be linked to the relevant fields specifically.
- Source-Subfield-Code und Target-Subfield-Code will not be analysed and are irrelevant!

8.9 mab2usm_902 - Processing mab-fields 902, 907, ... (Subjects)

This program processes the chain links of strings of indexing terms in field 902#, 907# to 947#.

The Target-Field-Code will be determined by the indicator of the first field inside a string of indexing terms (one or more 902-fields, one or more 907-fields,…):

"**902p** " → "**60014**"
"**902g** " → "**651 4**"
"**902s** " → "**650** "
"**902k** " → "**61014**"
"**902c** " → "**61014**"
"**902z** " → not permitted as1.field, no processing !
"**902f** " → not permitted as1.field, no processing !
"**902t** " → "**63004**"
"**902** " → not permitted as1.field, no processing !

The Source-Subfield $$a will be adopted as Target-Subfield $$a at the 1. field of the string. The Source-Subfield $$a of all following fields of a string, will be appended – subject to the indicator - as different Target-Subfields to the newly generated field:

Indicator "**p**" : **$$a** -> **$$x**
Indicator "**g**" : **$$a** -> **$$z**
Indicator "**s**" : **$$a** -> **$$x**
Indicator "**k**" : **$$a** -> **$$x**
Indicator "**c**" : **$$a** -> **$$x**
Indicator "**z**" : **$$a** -> **$$y**
Indicator "**f**" : **$$a** -> **$$v**
Indicator "**t**" : **$$a** -> **$$t**
Indicator " " : **$$a** -> **$$x**

In principle, the fixed positions and possible link numbers will always be removed from the Source-Subfield-Text von $$a before the adoption.

- Removing of the first 3 characters, if the third character is a pipe "**|**".
- Removing of the first 22 characters, if the third character is not a pipe "**|**".

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
902#            mab2usm_902
907#            mab2usm_902
912#            mab2usm_902
...
947#            mab2usm_902
```

- The Source-Field-Code has the standard meaning, but the current field code isn't analysed in the program!
- That's why the program has to be linked to the relevant fields specifically.
- The Target-Field-Code is determined by the program.
- Source-Subfield-Code und Target-Subfield-Code will not be analysed and are irrelevant!

8.10 mab2usm_700 - Processing mab-field 700 (Notation)

This program processes the notation fields.

At the moment, the only purpose of this program is to remove the first 20 characters from the (first) Subfield $$a, if the 1. character is not a pipe "|". Then, the first 20 characters will be interpreted as link number, which shouldn't be adopted. If the first character is the pipe "|", only this character will be removed!

After this modification of the Source-Field-Text the processing continues with the standard program edit_field, which can also be run with specific parameters.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
700   a 084   a mab2usm_700  parameter
700a  a 080   a mab2usm_700  parameter
.....
```

***parameter* :** as for program **edit_field** (see point **5.6**)

- The meaning of column 1 to 4 is equal to the standard described under point **4.1** .
- The processing of edit_field will be continued, exceopt from the above explanations. You may code specific parameters for that.

8.11 mab2usm_800 - Processing mab-fields 800, 802, ... (Secondary entries)

This program processes the person- and corporate body-fields of the secondary entries.

At the moment, the only purpose of this program is to remove the first 20 characters from the (first) Subfield $$a, if the 1. character is not a pipe "|". Then, the first 20 characters will be interpreted as link number, which shouldn't be adopted. If the first character is the pipe "|", only this character will be removed!

After this modification of the Source-Field-Text the processing continues with the standard program edit_field, which can also be run with specific parameters.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
800   a 7001  a mab2usm_800  parameter
802   a 7101  a mab2usm_800  parameter
806   a 7001  a mab2usm_800  parameter
808   a 7101  a mab2usm_800  parameter
812   a 7001  a mab2usm_800  parameter
814   a 7101  a mab2usm_800  parameter
818   a 7001  a mab2usm_800  parameter
820   a 7101  e mab2usm_800  parameter
824   a 7001  a mab2usm_800  parameter
826   a 7101  e mab2usm_800  parameter
```

***parameter* :** as for program **edit_field** (see point **5.6**)

- The meaning of column 1 to 4 is equal to the standard described under point **4.1** .
- The processing of edit_field will be continued, exceopt from the above explanations. You may code specific parameters for that.

9 Description of MAB-to-UNIMARC convit-programs

Overview

****** This re-formatting is not and respectively with special programs ******
****** supported at the moment!! ******

10 Description of USMARC-to-UNIMARC convit-programs

Overview

****** This re-formatting is not and respectively with special programs ******
****** supported at the moment!! ******

11 Description of UNIMARC-to-USMARC convit-programs

Overview

****** This re-formatting is not and respectively with special programs ******
****** supported at the moment!! ******